"From then on she sang with all her soul [...] When she began to invoke the angels and rise into the air, she took the whole quivering audience with her, and they all felt that they had wings."

– Gaston Leroux, *The Phantom of the Opera*, 1910. [1]

[1] Translated by Lowell Bair

University of Alberta

AUGMENTING LIVE MUSICAL PERFORMANCE THROUGH MUSIC VISUALIZATION

by

Robyn Taylor

©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2006

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-494-13894-7
Our file Notre référence
ISBN: 0-494-13894-7

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Dedicated to my family and friends, for all their love and support.

Abstract

We describe a framework used to generate real-time music visualizations that are used to augment a live musical performance. We present a musical feature extraction system implemented in the music processing environment, Max/MSP, where the perceptual parameters of pitch, loudness and timbre are extracted from a vocalist's performance, and chords played on a keyboard are identified. These extracted musical features are then mapped to responsive visual metaphors. We demonstrate the capabilities of our system through three example music visualizations. The first visualization uses Torres and Boulanger's ANIMUS Framework to illustrate emotional content in interpreted melodies through the behavioural responses of a virtual character. The second visualizes a vocalist's timbre through responsive video manipulation. The third illustrates vocal dynamics by manipulating visible aspects of an immersive virtual environment. This system has been used to create a multimedia performance piece, and is designed to facilitate future collaboration between artists and scientists.

Acknowledgements

I would like to thank Pierre Boulanger for all his support these past years. He has provided me with the opportunity to study and work in the most creative and dynamic laboratory environment I could ever imagine. I also want to thank Walter Bischof for all the help and encouragement he has always given me.

Thanks as well must go to the inspirational individuals I have met through the Banff New Media Institute, particularly Sara Diamond for providing me with the opportunity to participate in her incredible summits and events, and Maria Lantin for generously sharing her uniquely creative and technical skills.

Appreciative thoughts are also extended to my labmates in the Advanced Man-Machine Interface Laboratory as well as the students, faculty and staff of the Department of Computing Science.

My family and friends have been endlessly good to me, so I must thank Mum, Dad, Ryan, Grandma, Chris and Melanie for always being there.

Finally, I want to thank Frances Schuchard for instilling in me a lifelong love of music and a fascination with the Circle of Fifths.

Table of Contents

1 Introduction 1
- 1.1 Motivation . . . 2
- 1.2 Thesis Objectives . . . 3
- 1.3 System Description . . . 3
- 1.4 Real-Time Max/MSP Sound Feature Extraction Unit . . . 5
- 1.5 Three Music Visualization Implementations . . . 6
  - 1.5.1 Musically Responsive ANIMUS Characters . . . 6
  - 1.5.2 Illustrating Timbre using Responsive Video . . . 7
  - 1.5.3 Musical Control of a Reactive Immersive Environment . . . 7
- 1.6 Thesis Structure . . . 8
- 1.7 Thesis Contributions . . . 9

2 Fundamental Acoustical Concepts 10
- 2.1 Characteristics of Acoustic Sound . . . 10
  - 2.1.1 Pure Tones . . . 10
  - 2.1.2 Periodic Complex Tones . . . 11
- 2.2 Timbre . . . 15
  - 2.2.1 Spectral Envelope . . . 15
  - 2.2.2 Temporal Envelope . . . 16
  - 2.2.3 Vowel Formants . . . 17
- 2.3 Music-Theoretical Constructs . . . 18
  - 2.3.1 Hz to Note-named Pitch . . . 18
  - 2.3.2 Key . . . 19
  - 2.3.3 Chords . . . 19

3 Literature Survey 20
- 3.1 Extracting Features from Musical Input . . . 20
  - 3.1.1 Acoustic Sound Extraction . . . 21
  - 3.1.2 Melody Determination and Score Following . . . 24
- 3.2 Visualizing Musical Features . . . 27
  - 3.2.1 Virtual Character Music Visualization . . . 27
  - 3.2.2 Musically Responsive Immersive Art Installations . . . 30
- 3.3 Summary . . . 34

4 Extracting Features from Live Music 36
- 4.1 Overview of the Musical Perception Filter Layer . . . 37
- 4.2 Extracting Musical Feature Data from Real-Time Audio and MIDI Signals . . . 38
  - 4.2.1 Cycling '74's Max/MSP Environment . . . 38
  - 4.2.2 The Musical Perception Filter Layer Implemented as a Max/MSP Patch . . . 39
  - 4.2.3 Extracting Vocal Information Using the `fiddle~` Object . 41

4.2.4 Monitoring Keyboard Input 43
4.3 A Tonal Encoding Model for Musical Feature Organization 45
4.3.1 Western Tonal Music 45
4.3.2 A Music-Theoretical Melody Encoding System 47
4.4 System Architecture 49
4.4.1 Musical Input 49
4.4.2 Musical Feature Extraction 49
4.4.3 Musical Feature Organization 51
4.4.4 Network Communication using VRPN 51
4.4.5 The Visualization Engines 52
4.5 Summary of Contributions 53

5 Visualizing Music through Behaviourally Responsive ANIMUS Characters 54
5.1 Chapter Overview 55
5.2 Torres and Boulanger's ANIMUS Project 56
5.2.1 Perception Layer 57
5.2.2 Cognition Layer 58
5.2.3 Expression Layer 58
5.2.4 The *Alebrije* Character 59
5.3 Creating Musically Responsive ANIMUS Characters 59
5.3.1 Interfacing with the Musical Perception Filter Layer 60
5.3.2 Simulating an Emotional Response in the Cognition Layer . 61
5.3.3 Visualizing Character Emotions Through Movement 65
5.4 An Example of a Musically Responsive Character 66
5.4.1 Example: Greensleeves 67
5.4.2 Observations on Alebrije's Musical Responsivity 70
5.4.3 Summary of Contributions 72

6 Visualizing Music Using Responsive Video 74
6.1 Visualization Mechanism 75
6.1.1 Cycling '74's Jitter 75
6.1.2 Selected Video Manipulation Techniques 76
6.2 Visualized Parameters 80
6.2.1 Vocal Timbre 80
6.2.2 Visualizing Vocal Timbre 81
6.2.3 The Circle of Fifths 84
6.2.4 Visualizing the Circle of Fifths 85
6.3 The *Deep Surrender* Multimedia Project 86
6.3.1 Part One 86
6.3.2 Part Two 86
6.3.3 Part Three 87
6.3.4 Credits 89
6.4 Summary of Contributions 89

7 Visualizing Music Inside an Immersive Environment 91
7.1 Virtual Reality Applications in Immersive Environments 91
7.2 Virtools 92
7.2.1 The Virtools Development Environment 93
7.3 The *Phantom of the Opera* Visualization 96
7.3.1 Musical Control 96
7.3.2 Audio Visual Mapping Example 97
7.3.3 Future Development Plans 99
7.4 Summary of Contributions 100

8 Conclusion **102**
8.1 Music Visualization System Summary 102
8.2 Expressing Musical Emotion through Visualization 103
8.2.1 Extracting Musical Descriptors 103
8.2.2 Interpreting Emotional Content in Musical Performance . . 103
8.2.3 Illustrating Emotion Through Responsive Music Visualization 105
8.3 Future Artistic Direction 107
8.4 Future Applications 107
8.5 Summary of Contributions 108

Bibliography **110**

List of Figures

1.1 System Diagram 4

2.1 The Waveform of a Pure Tone, [27], p. 11. 11
2.2 Waveform of a Periodic Complex Tone (excerpted from [2], p. 18.) . 12
2.3 The Waveform of a Complex Tone Decomposed into Two Components [2], p. 18. 13
2.4 The Harmonic Spectrum [2], p. 20. 14
2.5 Waveforms and Spectra of Violin and Vibraphone [27], p. 41. 15
2.6 Amplitude Changes During Attack, Decay, Sustain and Release . . 16
2.7 The Amplitude of the Attack and Decay Phases of (a) a Cello and (b) a Piano [2], p. 13. 17
2.8 Waveforms, Spectra, and Formants of Vocalized Vowels [27], p. 218. 18

3.1 Characteristics of Several Musical Feature Extraction Systems . . . 22

4.1 Musical Perception Filter Layer 37
4.2 Musical Perception Filter Layer Max/MSP Patch 40
4.3 The `fiddle~` Object (as seen in the `fiddle~` help file) [28] . . . 41
4.4 Chord Detector Sub-Patch 44
4.5 An example of pitch encoding 48
4.6 System Architecture 50

5.1 The Musician Interacts with the Virtual Character, *Alebrije* 55
5.2 Integrating Musical Control into the ANIMUS Framework 61
5.3 Alebrije in two poses 66
5.4 The Singer Interacting with Alebrije 68
5.5 Greensleeves Melody 69

6.1 *Deep Surrender* Live Performance 75
6.2 Original Image x Mask Matrix = Modified Original Image 78
6.3 Replacement Image x Mask Matrix = Modified Replacement Image 78
6.4 The Resulting Chroma-keyed Image 79
6.5 Visualizing Tone through Colour 81
6.6 The Circle of Fifths 84
6.7 Mapping the Circle of Fifths to the Colour Wheel 85
6.8 Deep Surrender Part One 87
6.9 Deep Surrender Part Two 88
6.10 Deep Surrender Part Three 88

7.1 The University of Alberta VizRoom 92
7.2 The Phantom of the Opera Virtools Project 93
7.3 The Virtools Process Loop (adapted from [30], p. 97) 94
7.4 An Example of Visual Programming in Virtools [30], p. 143 95
7.5 Visualizing Pitch and Amplitude with Particle Fog 98

Chapter 1

Introduction

Each time a song is interpreted, the notes and words may be the same, but the experience is unique. Interaction between the vocalist and fellow musicians, the musicians and the audience, the energy of the room, and the adrenalin rush of performing all contribute to the serendipitous nature of live music, making each performance preciously ephemeral and distinct.

When I sing a piece, I know *what* I will be singing, but I like to give myself freedom to decide in the moment *how* it will be sung. Concert or theatrical productions which accompany live musical performance with pre-recorded visuals lack the flexibility to let the artist manipulate imagery as well as sound.

This is why, when visualizing a piece of interpretive music, it is important for the visualization medium to convey the nuances of the live performance. Each repetition of the visualization experience should be subtly unique, because no two live performances will ever be the same.

This thesis presents three methods of augmenting a live musical performance by mapping musical features to responsive imagery:

- visualization of emotional content through the behaviour of a virtual character
- visualization of vocal timbre using responsive video
- visualization of vocal dynamics inside a reactive immersive environment

1.1 Motivation

Music and visual content are often combined in popular culture. Dramatic scenes in films rely heavily on musical soundtracks in order to convey mood to the audience. Video game designers choose complementary music in order to heighten suspense, tension, or excitement during game play. When creating an artistic piece, we know that music and pictures can be used to tell a story or to create an atmosphere.

Too often, audio-visual multimedia is pre-determined, pre-orchestrated, and pre-choreographed. While this is a reasonable constraint for movies and pre-taped television programs since they have no interactive component, there are instances where it may be desirable for audio-visual content to be responsive to live input and real-time control. Live concerts and theatrical presentations are two examples of performance environments which would benefit from the additional flexibility and artistic freedom provided by a visualization system which can be interactively controlled as the performance develops.

Advances in virtual reality technology have facilitated the creation of a responsive medium that can be used for artistic expression. Three-dimensional graphics, large screens, and immersive technologies like the University of Alberta VizRoom allow an artist to draw her audience into a virtual environment that can be used to visualize the message she wishes to convey. Unlike pre-recorded audio-visual presentations where music and visuals must be combined in a static way, developments in real-time interactivity make virtual reality systems able to respond dynamically to cues in live performance. This allows visuals to be generated and manipulated 'on-the-fly'.

New Media art often combines modern technologies with traditional art forms to create new platforms for artistic expression. Virtual and augmented reality technologies can be used to visualize live music for the purpose of creating an artistic experience. Examples of existing music visualization artworks include immersive

installations that respond to vocal or instrumental input [24] [23], responsive virtual characters that synchronize their dancing to the rhythms played by musicians [31] [19], and concert performances that augment live performers' actions with responsive imagery [18]. Each of these visualization systems uses a different mapping between musical and visual content.

When using music visualization to augment the experience of observing a musical performance, artists must identify an appropriate mapping between sound and imagery in order to create an aesthetically pleasing multimedia environment.

1.2 Thesis Objectives

This thesis describes a framework which allows live music to be used as input. Through a set of filters, key parameters are extracted from live music. These parameters are used to activate and control various actions and responses in a virtual world. The main issues addressed are:

- How can information from live musical data be extracted and parameterized so that it may be used as an input system to a visualization engine?
- How can these extracted musical parameters be represented in a way that enables further processing based on music-theoretical analysis?
- How can a mapping be created between musical parameters and visual parameters for the purposes of creating an expressive artistic work?
- How can this framework be implemented in a rapid prototyping environment that an artist can use easily?

1.3 System Description

The proposed system is developed in a distributed fashion, allowing the tasks of musical feature extraction and analysis to be separated from the visualization system

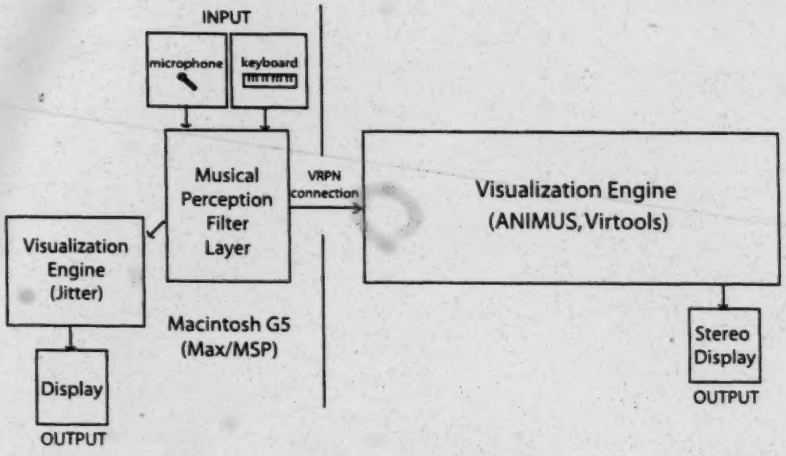


Figure 1.1: System Diagram

(see Figure 1.1). This facilitates the task of creating multiple mappings between music and imagery.

The input to the system comes from a digital keyboard and vocal microphone. The keyboard and microphone are connected to a Macintosh PC which handles the task of parameterizing musical input. Musical input is processed inside the Musical Perception Filter Layer, which is implemented using a visual programming environment called Max/MSP. Max/MSP is specially designed for music processing applications.

The parameterized input can be used to control three different visualization environments:

- **Jitter:** Control of responsive video parameters
- **ANIMUS:** Control of internal state of an avatar
- **Virtools:** Control of a responsive Virtual Reality environment

The Jitter video processing environment runs natively on a Macintosh, and is built into the Max/MSP processing environment. The ANIMUS and Virtools envi-

ronments run on remote PCs running the Windows XP operating system. Communication between the PCs and the Macintosh is performed using the Virtual Reality Peripheral Network (VRPN) networking library [35]. VPRN, developed at the University of North Carolina at Chapel Hill[1] is designed to facilitate generic distributed communications for virtual reality applications.

The output of all the visualization engines can be displayed upon large-scale projection screens. Additionally, the ANIMUS and Virtools simulation engines are capable of generating stereoscopic imagery.

1.4 Real-Time Max/MSP Sound Feature Extraction Unit

In order to visualize live music, it is essential that the stream of live music be parsed into discrete parameters. Cycling '74's sound processing environment Max/MSP [7] is an ideal development environment for this task, as it is designed to allow a programmer to easily manipulate audio and MIDI data. Max/MSP is a visual programming environment designed for music industry hobbyists or professionals with little or no formal training in computer programming.

Max/MSP allows musicians to create music processing applications by describing the dataflow between hierarchical submodules, known as *objects*. Its ease of use and modularity have made it an industry standard in electronic music development. Numerous users share their objects with a large Max/MSP user community.

The user-created Max/MSP object, `fiddle~`, developed by Puckette *et al.* [28] is used in our Musical Perception Filter Layer in order to extract pitch and amplitude from live singing. Additionally, we examine the raw harmonic data produced by `fiddle~` to create a descriptor of vocal timbre that a singer can control by modifying the vowel sound she sings.

[1]The use of the VRPN library was made possible by the NIH National Research Resource in Molecular Graphics and Microscopy at the University of North Carolina at Chapel Hill.

The Musical Perception Filter Layer also contains our own sub-module used to extract chord data from the keyboard. Keyboard data received in MIDI format is monitored and compared against a list of 'known' chords in order to determine what chords the user is playing.

Since the objective of this application is to facilitate the creation of music visualizations that are aesthetically pleasing to humans familiar with Western tonal music (the tonal system familiar to listeners of modern popular or folk music, as well as pre-twentieth century classical music), an existing schema for musical data representation which is congruent with the rules of Western music tonality has been modified in order to organize the extracted data.

The extracted musical feature data (vocal pitch, amplitude, and timbral information as well as keyboard chord data) is then transmitted to the visualization engine.

1.5 Three Music Visualization Implementations

This thesis presents three implementations of mappings between music and visuals. In each visualization implementation a different mapping is used to illustrate properties of interpreted music.

1.5.1 Musically Responsive ANIMUS Characters

In this implementation a virtual character created using Torres and Boulanger's ANIMUS Framework [38] [37] is used to visualize the emotive content of music by expressing simulated emotion through responsive behaviours. In order to assess and identify emotional signifiers in interpreted music, a study by Deryck Cooke [5] correlating melody and composers' emotional intentions is explored. Aspects of Cooke's research serve as the basis for an ANIMUS character's cognition mechanism in this implementation. Cooke associates emotive meaning to the tonal structure of Western melodies, associating each tone in the musical scale to an emotional

context. We implemented a virtual character who could interpret melodies in a way consistent with Cooke's metric.

1.5.2 Illustrating Timbre using Responsive Video

Cycling 74's Jitter [6] is a video processing package that can be integrated into the Max/MSP environment. The Jitter package allows users to create responsive video applications by describing (using visual programming) how data flows between Max/MSP and Jitter *objects*. Jitter can be used for a variety of visualization tasks such as manipulation of video playback, generation of basic 3D animation, or the modification of still images. We have used the Jitter environment to create a visualization which illustrates vocal timbre and harmonic relationships between key signatures by layering video clips and adjusting colour balance. This visualization system was used to create an interactive piece called *Deep Surrender*, that has been performed live in a concert setting.

1.5.3 Musical Control of a Reactive Immersive Environment

The Virtools environment [8] is a visual programming environment which allows designers of virtual reality applications to create immersive visualizations. A musical control system for Virtools applications has been created, using VRPN to connect music extraction routines to behaviours inside the Virtools environment. This implementation allows a vocalist to use his or her voice to formulate particle clouds generated by Virtools' particle generation and interaction routines. Virtools' visual programming environment allows different visualization metaphors to be easily defined, tested, and modified. Connecting our musical feature extraction system to the Virtools environment allows us to rapidly develop music visualization applications. This illustrates one of the benefits of our proposed architecture: the connection between Max/MSP's music processing environment and Virtools' virtual reality simulator allows both the musical and visual aspects of immersive music visualization

projects to be implemented using visual programming techniques.

1.6 Thesis Structure

In Chapter 2, background information is provided which explains acoustical concepts and musical terms that should be understood when reading this thesis.

In Chapter 3, a survey of the state-of-the-art in this field is presented. Related musical feature extraction techniques are discussed. Applications which use virtual characters to illustrate musical features, and artistic installations incorporating music visualization technologies are described and analyzed.

Chapter 4 of this thesis presents the musical feature data extraction and representation system used in this implementation. An overview of the Max/MSP development environment is provided. The methods used in the extraction of musical information are presented (including an overview of the technique used by Puckette *et al.* [28] in the `fiddle~` pitch tracking object). In addition, the schema used to represent tonal melodies is described.

Chapters 5 to 7 discuss the music visualization projects that have been created using the proposed framework. Chapter 5 describes the musically responsive ANIMUS character, and presents a discussion on how Deryck Cooke's research [5] can be used to infer emotional intention from interpreted melodies. Chapter 6 describes the *Deep Surrender* responsive video project in which vocal timbre and harmonic relationships are used to control video playback in a live performance setting. Chapter 7 discusses musical interactivity inside the Virtools immersive environment, and describes the connection between Max/MSP and the Behaviour Engine that forms the core of the Virtools simulator.

Chapter 8 summarizes the contributions made in this thesis and briefly discusses future research avenues.

1.7 Thesis Contributions

In this thesis, we have made several contributions to music visualization research, such as:

- A new architecture facilitating the creation of music visualization applications has been created. Musical feature extraction tasks are programmed in the visual programming environment, Max/MSP. Extracted data is represented in a tonal context.
- Using our framework, three ways of expressing musical content using visual parameters are presented:
  - Analyzed musical features are used to interact with a virtual character whose cognitive system is defined using Cooke's metric to determine its emotional responsivity to music.
  - The music visualization framework has been used to create and perform an original multimedia piece that visualizes aspects of a vocalist's timbre.
  - Musical features extracted in MAX/MSP can be used to interact with the visually programmed virtual reality simulator, Virtools.

To illustrate our contributions, we have implemented three working prototypes which demonstrate the capabilities of our approach.

Chapter 2

Fundamental Acoustical Concepts

The information in this chapter summarizes material contained in texts by Plack [27], Campbell and Greated [2], Pierce [26], and Sundberg [32]. The information is provided for completeness and for readers who are not familiar with certain musical concepts. The terms defined in this Chapter will be used throughout the thesis.

2.1 Characteristics of Acoustic Sound

What we call 'sound' is the result of waves of air pressure fluctuations reaching our ears. If we wish to examine the characteristics of a sound, we must examine the sound's waveform, which visualizes the way air pressure varies over time.

2.1.1 Pure Tones

Before considering complex sounds, let us examine the fundamental properties of sound using the simplest type of sound wave: a pure tone, also known as a sine wave. This is the type of sound generated by a tuning fork, or by an oscillator in a laboratory.

A pure tone is generated by air pressure rising and falling sinusoidally. Figure 2.1 shows a pure tone waveform

Amplitude describes the maximum amount by which the sound wave varies from its average value. **Frequency** is defined as the number of times the sound wave

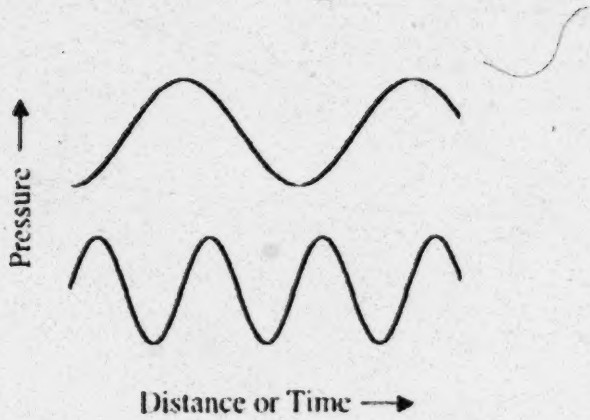


Figure 2.1: The Waveform of a Pure Tone, [27], p. 11.

completes its cycle within a given time frame. Frequency is generally measured in cycles per second, known as hertz (Hz). The **period** of a sine wave is the inverse of frequency, and describes the length of time it takes the wave to complete one cycle. A sine wave's **phase** describes the point on the cycle that the sine wave has reached at a specified time.

When visualizing music in order to create an artistic application, we are concerned with the audience's perception of the musical information. While frequency and amplitude describe physical parameters of a pure tone, pitch and loudness describe their perceptual counterparts. When we refer to the **pitch** of a pure tone, we are referring to the tone's frequency. By **loudness**, we mean the perceptual sensation caused by the tone's amplitude.

2.1.2 Periodic Complex Tones

The waveforms of natural sounds are more complex than the waveforms of pure tones. Figure 2.2 shows a complex waveform.

Musical sounds (with the exception of percussive instruments) and vocalized vowel sounds are examples of periodic complex tones. They are not sinusoidal, yet

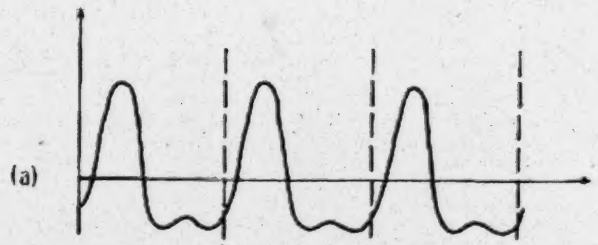


Figure 2.2: Waveform of a Periodic Complex Tone (excerpted from [2], p. 18.)

they share the property of periodicity with pure tones – they repeat over time at the rate of one cycle each period.

The **fundamental frequency** (in Hz) of a periodic complex tone is equal to the number of times the waveform cycles each second. This can be considered equivalent to the **pitch** of the sound. The **amplitude** of the complex tone describes the maximum value reached during the tone's cycle. In perceptual terms, this would be considered the **loudness** of the sound.

Harmonics and the Fourier Transformation

According to Fourier's theorem, any periodic wave is equivalent to a summation of sinusoidal waves, each sinusoidal wave having specific amplitude, frequency and phase. The frequency of each sinusoidal wave, or Fourier component, that comprises a complex wave is equal to an integer multiple of the fundamental frequency of the complex wave. A periodic waveform of any complexity can be represented in this manner if enough sine waves are added to the representation.

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx \tag{2.1}$$

The Fourier transform (see Equation 2.1) is used to decompose a periodic wave into its sinusoidal components [1].

The sinusoidal components that form a complex sound are called **harmonics**, or **partials**. The second harmonic frequency is twice the frequency of the fundamental

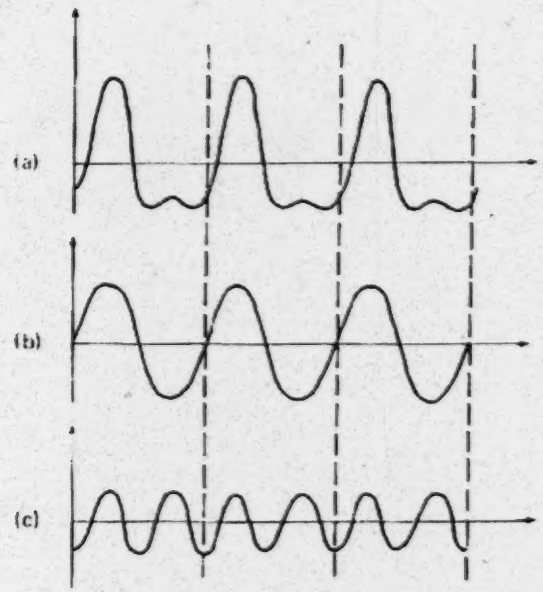


Figure 2.3: The Waveform of a Complex Tone Decomposed into Two Components [2], p. 18.

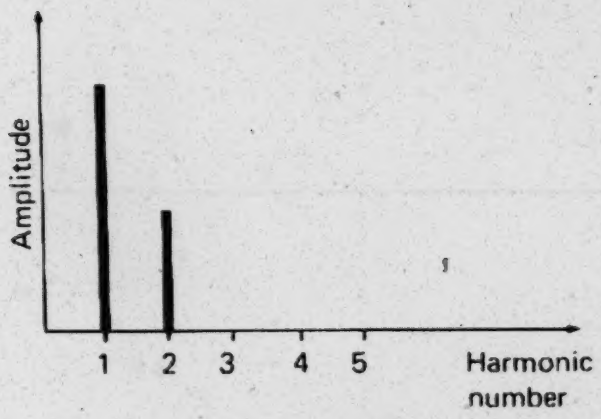


Figure 2.4: The Harmonic Spectrum [2], p. 20.

frequency, the third harmonic frequency is three times the fundamental frequency, etc.

Figure 2.3 shows a complex wave decomposed into its two components: a sine curve with a frequency equaling the fundamental, and a sine curve with a frequency equaling twice the fundamental and an amplitude that has been reduced by half.

Spectral Representation

When we represent a periodic complex sound as a waveform, we see it as a function of time. This does not allow us to easily see the amplitude of each of the sound's harmonics. If we wish to view the harmonic components of a periodic complex sound, we can represent it in the form of a **spectrum**. The harmonic spectrum plots the amplitude of each of the sound's harmonics at each of of the harmonic frequencies. The spectral representation does not show us how the sound changes over time, but rather shows the amplitude of the harmonic components of the sound at a specific time.

Figure 2.4 shows the harmonic spectrum of the waveform displayed in Figure 2.3. Each vertical bar represents the amplitude of a harmonic.

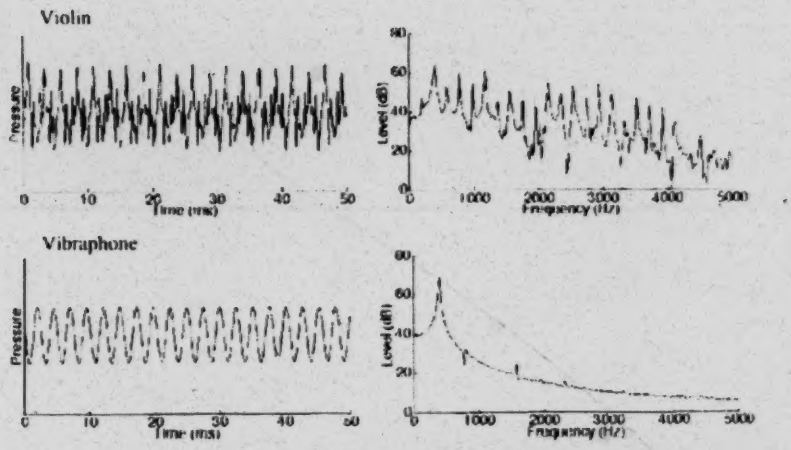


Figure 2.5: Waveforms and Spectra of Violin and Vibraphone [27], p. 41.

2.2 Timbre

The **timbre** of a sound describes "that property which permits it to be distinguished from another sound of the same pitch and loudness." [2], p. 142. The spectral and temporal envelopes describing the sound contribute to the sound's unique timbre. In vocalists, timbre is also influenced by vowel production.

2.2.1 Spectral Envelope

The **spectral envelope** of a sound contains the amplitudes of a sound's harmonics. The relative amplitudes of the harmonics are what makes the sound of a violin different from a that of a vibraphone, even if they are both being played at the same pitch and loudness.

The way energy is distributed amongst the harmonic frequencies of each sound determines its timbral character.

Figure 2.5 compares the differences in the waveforms and spectra of a violin and vibraphone. A vibraphone has a hollow ringing tone, while a violin's tone is bright and rich.

The spectrum of the violin's sound shows high amplitudes at many of the partial

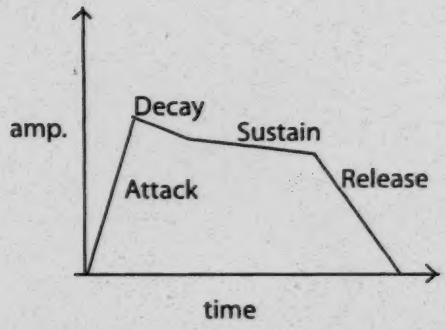


Figure 2.6: Amplitude Changes During Attack, Decay, Sustain and Release

component frequencies. The spectrum of the vibraphone's sound shows a high amplitude at the fundamental, with less amplitude at each successive higher partial. The vibraphone's sound would be described as having less **brightness** than the sound of the violin. This is because the violin's harmonic spectrum shows a high concentration of amplitude at higher partials.

2.2.2 Temporal Envelope

Timbre is also affected by the **temporal envelope** of a musical sound. The temporal envelope refers to the way that a sound's characteristics change over time. A musical sound does not sound consistently the same during all phases of its existence.

The sound can be broken into several distinct phases (see Figure 2.6.)

- Attack (or onset) phase – the sound builds up to its intensity during the attack phase
- Decay – the initial burst of intensity subsides
- Sustain – after decaying, the sound reaches its steady sustained level
- Release – when the musician stops playing, the sound level falls

The amplitude patterns seen in the Attack, Decay, Sustain and Release phases of a particular instrument help define that instrument's timbre. The harmonic com-

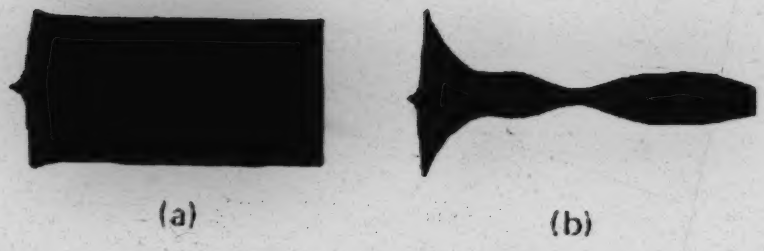


Figure 2.7: The Amplitude of the Attack and Decay Phases of (a) a Cello and (b) a Piano [2], p. 13.

ponents that create the instrument's sound may also vary during the various phases of sound production, further defining the timbral characteristics of the instrument.

Figure 2.7 illustrates how amplitude changes in the attack and decay phases of two different instruments. As we can see, the cello's attack and decay phase is more gradual than the dramatic attack and decay seen in the piano diagram. This is consistent with the nature of the instruments, as the bowing action used to generate sound from a cello is much less intense than the hammer action that produces sound from a piano.

2.2.3 Vowel Formants

Vowel sound production affects the timbral characteristics of the human voice. Our vocal tract is a **resonator** which we can manipulate using muscular control. Adjustments made to the musculature of our vocal tract, jaw, and tongue affect the placement of its resonant regions. Vocal resonant regions are areas of the instrument (the vocal tract) which resonate intensely at specific frequencies.

If we examine the spectra of two vocalised vowels (seen in Figure 2.8) we will see that the harmonic spectrum of each vowel sound is characterised by prominent peaks. These peak regions are known as the **formants** characterizing the vowel.

Adjusting the placement of vocal resonators is what generates vowel sounds and modifies a singer's timbre. When a person makes adjustments to the positioning of

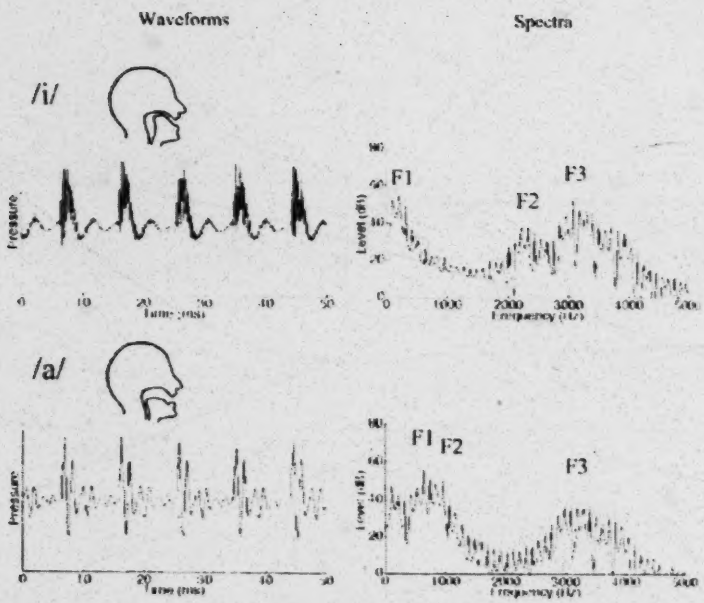


Figure 2.8: Waveforms, Spectra, and Formants of Vocalized Vowels [27], p. 218.

their vocal tract, jaw, and tongue in order to vocalize a desired vowel, the vocal resonators are being manipulated so that their resonances are consistent with the formant regions which characterize the desired vowel. The harmonic spectrum of a vocalized vowel will then have the observable formant structure that defines the vowel.

2.3 Music-Theoretical Constructs

Several music-theoretical terms are used in this thesis.

2.3.1 Hz to Note-named Pitch

Instead of representing pitches in terms of Hz, the convention in music theory is to signify pitches by note names. The frequency 440 Hz is equivalent to the note name A4 (the A above middle C).

A is the **pitch class** of the note, and 4 refers to the **octave** in which A4 appears. A4, A5, and A6 are all part of the same pitch class, but C4, D4, E4, F4, G4, A4, and B4, are all part of the same octave.

An **interval** of an octave is equivalent to a doubling in frequency. A4 is equivalent to 440 Hz, and A5 (an octave above A4) is equivalent to 880 Hz.

2.3.2 Key

Tonal music is defined to be in a particular **key**. The key of a piece determines the **tonic note** of the piece. The tonic note for both C major and C minor would be C.

2.3.3 Chords

Polyphonic instruments (like pianos) are capable of sounding more than one note at a time. When three or more notes are played simultaneously, they form a **chord**.

Major Chords contain three notes: the tonic note of the chord, a major third (four semitones) above the tonic, and a perfect fifth (seven semitones) above the tonic. **Minor Chords** contain the tonic note, a minor third (three semitones above the tonic) and a perfect fifth (seven semitones) above the tonic.

When a chord is played in such a way that the tonic is the lowest pitch, the major or minor third is the middle pitch and the perfect fifth is the highest pitch, it is being played in **root position**. A chord is being played in an **inversion** if the lowest pitch in the chord is the major or minor third or the perfect fifth.

Chapter 3

Literature Survey

The task of developing an effective music visualization metaphor requires that two important questions be addressed:

- What features (pitches, chords, etc.) are present in a piece of music?
- How can we illustrate these features using visual imagery?

This chapter is organized in two sections. In the first section, several methods of musical feature extraction are discussed, including techniques suitable for analysis of acoustic input as well as techniques for identifying melodic features for the purposes of score-following.

In the second part of this chapter, a survey of music visualization strategies is provided, discussing strategies that map sound to virtual character behaviour and strategies that combine real-world and virtual world imagery to enhance the visualization of musical sound.

3.1 Extracting Features from Musical Input

In order to extract control parameters to which responsive visualization can be mapped, it must be determined what musical parameters constitute important descriptors of the performance experience, and how these musical parameters can be extracted for further use.

Signal processing can be used to segment an acoustic stream of musical input into the perceptual parameters of pitch, loudness, and timbre (see Chapter 2).

Further analysis on extracted pitch data can be done in order to perform music-theoretical analysis on the musical input. Once the collection of input pitches is known, streams of pitches can be treated as musical melodies that can be matched against known scores for the purposes of score-following.

The following sections will survey techniques used for parameterizing acoustic sound and for relating extracted pitch data to known musical scores.

3.1.1 Acoustic Sound Extraction

Music generated by voice, piano, wind instruments or other acoustic instruments can be parsed and analyzed in order to identify and parameterize information about the performance. This musical feature data can then be used to control subsequent computer processing.

There exist numerous examples of techniques used to extract musical feature data from acoustic streams. Musical feature extraction systems that operate in real-time allow musical features to be used as interactive control mechanisms in entertainment or performance applications.

Figure 3.1 compares the musical analysis capabilities of three sound analysis toolkits. Each of these analysis engines have been used as control systems for artistic installations or augmented performances.

Puckette, Apel and Zicarelli's pitch tracker, `fiddle~` [28], estimates the pitch and amplitude of sound, provides notification when the onset (the attack) of any pitched or non-pitched sound is detected, and outputs raw spectral data describing the frequencies and amplitudes of each partial in the sound's harmonic spectrum. `fiddle~` is implemented using the music processing environment, Max/MSP [7]. `fiddle~` is widely-used by Max/MSP users, as it is freely available, effective at

Application	Pitch Tracking	Amplitude/ Loudness Monitoring	Identified Timbral characteristics	Additional Output Data
Puckette, Apel and Zicarelli's `fiddle~`	Yes	Yes	None	Onset notification for pitched and non-pitched events List of spectral peaks and associated amplitudes
Metois' digital signal processing toolbox `libtsd.a`	Yes	Yes	Brightness Pitch Ambiguity/ Noisiness	Customized for specific projects (formant tracking, harmonic analysis, etc...)
Jehan and Schoner's `analyzer~`	Yes (using `fiddle~`)	Yes	Brightness Pitch Ambiguity/ Noisiness	Spectral information about the sound data

Figure 3.1: Characteristics of Several Musical Feature Extraction Systems

detecting pitch, and easy to integrate into Max applications. It is used to form the pitch detection module inside Jehan and Schoner's `analyzer~`[14] sound analysis system.

Metois' [22] and Jehan and Schoner's [14] applications also extract pitch and loudness information, and additionally provide information about the timbre of the musical sound being produced. Metois' [22] signal processing library is capable of extensive audio analysis, and functions as the audio processing core for many of the musical interfaces and performance applications produced at the Massachusetts Institute of Technology. Metois' research defines a musical sound model that parameterizes the characteristics of a musical sound in terms of the sound's *pitch, loudness, brightness, and noisiness*.

While pitch and loudness are standard measurements associated with musical sound, Metois' quantization of brightness and noisiness represent an attempt to identify the timbral properties of the sound. Metois describes a sound's bright-

ness as a measure of how a sound's energy is distributed amongst the partials in the sound spectrum. Brighter sounds would have more energy distributed amongst high frequency partial components. Noisiness is described as the ambiguity between the estimated pitch reported once for each specified time interval and the actual non-discrete pitch data which fluctuates continuously. Since estimation is required to convert acoustic sound data into a stream of discrete pitches, detail describing the small fluctuations in the signal data is necessarily lost in the discretization process. Metois considers this detailed data 'noisiness', and argues that it is a relevant timbral measure. Metois' parameterization of sound has been used in a variety of MIT Media Lab's augmented performance projects. In particular, MIT's Brain Opera used Metois' parameterization of sound to control the real-time music visualization, *The Singing Tree* [23].

Jehan and Schoner's `analyzer~` [14] application implements aspects of Metois' signal processing methodology in Max/MSP. Their research identifies a timbral measurement that is additional to the extracted characteristics of pitch, loudness, brightness. They consider pitch, loudness and brightness to be deterministic components of musical sound. If the effects of these deterministic factors are isolated and removed from the harmonic spectrum describing the sound, the resulting 'residual spectrum' describes the noise characteristics which provide additional information about the timbre of the sound.

Jehan and Schoner's timbral measurements are primarily used to synthesize the retargeting of music from one acoustic instrument to another, but their `analyzer~` object has been used for augmenting live performance as well. In the orchestral work, *Sparkler* [13], `analyzer~` was used to measure timbral properties of the polyphonic sound generated by an orchestra. The resulting data was used to augment the performance by generating synthesized sound "clouds" that provided real-time responsive accompaniment to the orchestra.

Acoustic Sound Extraction within our Application

Our application uses Puckette *et al.*'s `fiddle~` object to extract pitch and amplitude information from acoustic (vocal) musical input. Chapter 4 of this thesis describes how `fiddle~` is incorporated into our Musical Perception Filter Layer.

In addition to pitch and amplitude information, our application makes use of the spectral peak information provided by the `fiddle~` object in order to make an assessment of vocal timbre. We choose to measure a different aspect of timbre than those measured by Metois' and Jehan and Schoner's signal processing applications. We use the harmonic data provided by `fiddle~` to assess vowel production in sung vocalization. Incorporation of Metois' and Jehan and Schoner's measures of 'brightness' and 'noisiness' could provide additional timbral information to our application, and should be considered as a possible avenue of future development.

3.1.2 Melody Determination and Score Following

Once acoustic sound streams are parsed, the resulting fragmented pieces of feature data describing the musical performance still lack music-theoretical integrity. Singers do not generally sing strings of dissociated pitches. They sing melodies. They sing phrases. Their vocalizations form cohesive *musical gestures*.

Pitch and amplitude information alone provide useful data about a performance that can be translated into knowledge about the musical gestures being performed.

If these musical gestures can be recognized, it is possible to compare live musical input with a known score, so that applications can be devised with a known script in mind, yet still be responsive to the variations inherent in live performance. Specific cues in the score can trigger appropriate visual or audible effects that complement the live performance. This enables the creation of augmented musical applications such as visualizations or generative accompaniments that are responsive, but also predictable.

McNab *et al.* describe a signal processing system [20] specifically designed for use with score-following applications [21]. They extract pitch and amplitude information from a stream of vocal input, and use this information to segment the analogue stream into discrete notes. Solo singing is monophonic, so one note must end before another one begins. They describe both pitch-based and amplitude-based methods used in the discretization of musical input streams.

Amplitude-based segmentation infers note separation when the amplitude of the input stream falls below a defined level. To facilitate this process, singers insert plosive consonant sounds ('ta' or 'da') between notes. The formation of these plosive consonants causes the amplitude of the sound stream to momentarily drop, allowing the amplitude tracker to distinguish individual notes. Pitch-based separation is achieved by monitoring the vocal input stream and reporting the occurrence of a note each time a pitch remains within a certain threshold for a certain amount of time. Contiguous pitch regions are assumed to be notes of extended duration.

Once the stream is parsed into notes, the melodies represented by the note sequences can be used for pattern-matching against known scores. McNab *et al.* pattern-match melodies extracted from sung music for the purpose of vocally querying melodic databases.

Error Correction Strategies

The imperfect nature of human performance causes score-following to present a complicated challenge: How can live input generated by error-prone humans be pattern-matched against machine-perfect scores? Singers don't always sing exactly on pitch. Pianists make small mistakes. Humans can still recognize the musical gestures they are performing (assuming the errors they make are sufficiently small), but how can a machine interpret such ambiguous data?

The difficulties caused by the ambiguity of human performance are heightened

in real-time score-following situations, when time is a critical factor in application success.

McNab *et al.* address the problem of missed notes and faulty pitches by pattern-matching between the musician and the score using melody contours instead of absolute pitches [20]. Melody contours encode melodies in terms of the intervals they contain rather than in terms of the pitches they contain. This allows a melody to begin on any chosen note. They also "forgive" intonation (pitch) errors by implementing a system of adaptive tuning which attempts to follow the user's pitch (if she sings consistently flat, the system constantly tries to adapt its melody contour tracking to accommodate her gradually descending tuning) [21].

Puckette and Lippe describe a method of increasing the robustness of score-following for the purpose of making it reliable for use in of real-time musical augmentation applications [29]. To help their score-following applications function despite the wrong notes that inevitably occur in live performance, Puckette and Lippe maintain a 'skip list' of any notes in the score that the performer does not execute in the expected sequence. Their pattern-matching algorithm gives these skipped notes preferential priority for subsequent matches. Having a bias towards matching missed note sequences helps avoid false matches that cause the pattern-matcher to jump ahead in the score and lose all hope of re-synchronizing with the performer.

Melodic Determination within Our Application

Our musical feature data extraction routines use the existing behaviours provided by the `fiddle~` object [28] in order to segment incoming musical data into its 'notes'. `fiddle~` uses a pitch segmentation strategy that is similar to both the pitch-based and amplitude-based segmentation systems described by McNab *et al.*[20]. `fiddle~` reports the occurrence of a new pitch when the amplitude of the pitch stream spikes by a predefined amount, or when the pitch stream's funda-

mental frequency increases or decreases by a semi-tone.

In the current implementation of our musical feature data extraction system we do not incorporate score-following capabilities, nor do we incorporate error correction strategies addressing the imprecision of human input. We would like to expand our system in the future to facilitate score-following (as described by Puckette and Lippe [29]) and adaptable tuning (as described by McNab *et al.* [21]) in order to increase system functionality and robustness.

3.2 Visualizing Musical Features

Once the stream of live music has been analysed and information describing interesting musical features has been extracted, a compelling visual representation of the musical information can be created. A successful visualization needs to illustrate aspects of musical performance by correlating musical features to an appropriate visual metaphor.

Visualizations may be created in a multitude of ways. Extracted musical feature data (chords, timbre, pitch, amplitude, etc.) can be mapped to any aspect of a virtualized environment in order to produce visible feedback in response to musical input.

Of particular relevance to this thesis are visualizations that illustrate musical features through the actions of virtual characters, as well as visualizations that use immersive technologies to transport audiences and participants into virtually augmented performance spaces.

3.2.1 Virtual Character Music Visualization

There exist numerous music visualization applications that use the metaphor of responsive virtual character behaviour to illustrate the interaction between music and imagery.

Much of the research addressing the correlation of music and animated character imagery deals with the generation of virtual character animations that are synchronized with the rhythmic constraints of music. Being able to automatically synchronize character movements to music is desirable, as it simplifies the process of creating the type of animations often seen in films and television.

Kim *et al.* [12] describe a rhythmic-motion synthesis system which enables the automatized generation of vast ballrooms full of digital characters waltzing in tempo with orchestral soundtracks. They create unique animations for each waltzing couple by traversing a pre-created motion synthesis graph. Inside this graph, numerous captured motions are segmented into kinematically constrained *motion beats* that can be synchronized with the rhythmic patterns in music. Traversing this graph in a randomized fashion for each character allows for the generation of natural looking unique animations that remain in synch with the musical soundtrack.

Cardle *et al.* describe an approach for animating virtual characters using musical interaction [3]. Their system extracts musical feature data from MIDI and analogue input and uses it to shape the motion curves that form character animations. They perform analysis upon musical input and extract chord, rhythm and pitch information, which they then use to modify character movements. They use rhythmic information to synchronize the timing of the animations, and monitor pitch and chord fluctuations in order to attempt to visualize musical events (such as major to minor key modulations) with appropriate animated responses, and to associate repetition in musical input with repetition in animated character movement.

Goto's beat tracking system [11] coordinates the rhythmic motions of *Cindy the Virtual Dancer* [19] with tempo information extracted from analogue musical signals. *Cindy* has predefined dance routines, the timing of which are adjusted to comply with the beat information Goto extracts. *Cindy*'s rhythmically synchronized dance movements are used as a visual feedback system to facilitate virtual jam ses-

sions. Her movements are controlled by the remote musicians and serve as a form of visual time-keeping. She is programmed to associate "moods" with different types of rhythmic improvisation. Her dance changes to reflect the mood of the jam session.

The Improv system created by Singer et al. [31] also allows virtual characters to associate different types of dancing with different "moods" conveyed in musical input. Their *Dancing Gregor* character also infers mood from rhythm. His movements are synchronized to the tempo of incoming drum beats, and his dancing style is chosen based on the rate and volume of the user's drumming. The Improv architecture separates the virtual character's internal state from his externally visible behaviour. Dancing animations are dynamically modified to express the virtual character's responses to the drummer.

Torres and Boulanger's ANIMUS Framework [37][38][36] similarly separates the virtual character's visible expressions from its internal cognitive state when formulating responses to events (including musical events) in the virtual world. A musically responsive ANIMUS character [33][34]) can respond to the emotional significance of the music-theoretical structure of a live musical performance. This is described in detail in Chapter 5 of this thesis.

Comparison of Virtual Character Music Visualization Techniques

While Kim *et al*'s rhythmically synchronized virtual characters [12] provide efficient and effective virtual characters suitable for the simulation of choreographed formations (dances, marches, etc.) or crowd scenes, they do not address the generation of virtual characters who appear to be cognizant of non-rhythmic musical subtleties. Their characters are graphically convincing, but do not display unique personalities or behavioural responses. Likewise, the animated characters described by Cardle *et al.* [3] do not simulate any cognitive awareness of the 'mood' contained

in a musical passage, although their animations are designed to vary in response to musical constructs like chord data, pitch, and rhythmic information.

Cindy the Virtual Dancer [19] evidences a rudimentary understanding of 'mood', which she visualizes through her dance styles. Her awareness of 'mood' in music is simulated through rule-based responses, however, and does not attempt to simulate any cognition process that is responsive to musical input.

The Improv system described by Singer *et al.* [31] and Torres and Boulanger's ANIMUS Framework [38] [37] represent more advanced strategies for responsive virtual character simulation. ANIMUS and Improv characters maintain cognitive states that can be affected by data in the virtual environment. The simulated 'mood' of the Improv character, *Dancing Gregor* is responsive to percussive intensities played by a live drummer. *Gregor*'s dancing style is influenced by his mood.

Our Musically Responsive Virtual Characters

The virtual character visualization described in this thesis presents real-time musically responsive ANIMUS characters [33] [34] who simulate a cognitive awareness of the emotive content of music. They generate believable response behaviours to illustrate their cognitive state. The musically aware ANIMUS characters constitute an advancement in the field of musically responsive virtual character visualization, as their simulated cognitive awareness of the association between music-theoretical rules and emotive content provides a novel way to illustrate the emotional properties of melody.

3.2.2 Musically Responsive Immersive Art Installations

Artists can use immersive technologies (large-scale stereoscopic displays, CAVEs, etc.) to draw audiences inside a compellingly realistic virtual world. Life sized and three dimensional responsive imagery can be used to blur the boundary between the physicality of the interactor and the virtual world within which he or she is im-

mersed [1]. Numerous artists have exploited the immersive properties of high-end visualization systems in order to create music visualization installations that incorporate the physicality of the performer, the audience, and the installation environment. These visualization schemes allow music to be tied to imagery that physically surrounds the participant, enhancing the synaesthetic experience of viewing sound.

Jack Ox's visualizations within an immersive CAVE system explore the harmonic structure of musical input [24]. Her *Color Organ* allows viewers to visualize relationships in a musical piece by creating three-dimensional structures and landscapes that observers can explore and navigate at will. The visualization engine allows different visualization schemes (which she terms 'stops' in keeping with her *Organ* metaphor) to be used in conjunction with each data set. To represent harmonic structure in music, Ox devised a 'stop' which maps a visual colour wheel with the music-theoretical device *The Circle of Fifths*. Each chord present in the musical input triggers a colour representing the chord's position mapped on the colour wheel.

The Singing Tree [23], created by Oliver *et al.* at MIT's Media Laboratory as part of MIT's *Brain Opera*, immerses a user inside an artistic space comprised of computer graphics displays and installed set pieces. In this installation, the user not only views a visual representation of sound, but he or she is able to manipulate the visualization by singing. The environment is visibly and audibly responsive to the sound of the user's voice. The participant's vocalization is analyzed using Metois' analysis engine [22], and the extracted parameters are used to modify the audio-visual multimedia surrounding the participant.

The Singing Tree's audio-visual feedback prompts a user towards the goal of singing a prolonged note at a steady pitch. The feedback system leads the user towards that goal by providing him or her with aesthetically rewarding multimedia

[1] Portions of this discussion appear in Taylor, Boulanger and Torres, 2005. Proceedings of the 5th International Symposium on Smart Graphics. Springer LNCS, pages 13-24.

as his or her performance improves and audio-visually dissonant multimedia if his or her pitch control degrades. The video feedback component progresses or reverses to and from a goal state (one of the videos has a flower which blooms if the user sings steady pitches and reverses into a bud if the pitch wavers). The auditory feedback is also goal-driven. Music is generated that contains angelic harmonies when the user produces a steady pitch, and dissonant brass and percussion sounds when the user's pitch becomes unsteady.

Oliver *et al.* took particular consideration when designing the *Singing Tree*'s auditory feedback system to make the feedback neither fully deterministic nor completely random. Repetitive use of the system generates sounds that are similar but not identical to the sounds previously generated. This indirect control strategy helps maintain the user's interest level, as he or she experiments with control over the system.

Elle et la voix [4], created by Ikam and Fléri at the Institut de Recherche et Coordination Acoustique/Musique (IRCAM) is another example of an installation that generates audio-visual multimedia in response to participant vocalization. *Elle* is represented as a giant female face, floating inside a room. When users enter the room, motion sensors track their positions, and *Elle*'s attention turns to them in acknowledgement of their presence. If they speak, *Elle*'s voice modulates (using a vocal resynthesizing module) to mimic aspects of their voices. Her voice matches their vocal intensity, the speed at which they are speaking or singing, and the volume at which they are speaking. She moves in response to the participants' movements and sounds, drifting through the room and echoing the sounds of their voices.

An audio-visual concert performance piece, *Messa di Voce*, created by Golan Levin and Zachary Lieberman [18], visualizes the vocalizations of live performers. Vocalizations are visualized as abstracted glyphs which originate from the locations of the performers' mouths. The concert is performed much like a traditional con-

cert, however, technology is used to augment the performance. Vocalists stand on a simple stage when performing the work. The stage's backdrop is comprised of giant screens, upon which the performance's augmented visualization is projected. The scale of the imagery makes the vocalists appear to their viewing audience to be a part of the virtual space within which they are performing. The glyphs generated from the vocalists' musical sounds become as much a part of the virtually augmented world as their physical bodies, providing audiences with the illusion that the physical and the virtual are equally real.

Interactivity in Immersive Music Visualization

The previously described installations illustrate how virtual spaces can be used as artistic environments within which music can be visualized. Each installation provides a different example of how the real and the virtual can be combined for the purposes of artistic expression. The installations differ in the ways the audience is permitted to interact with and observe the space.

Ox's *Color Organ* [24] uses a CAVE environment to visualize musical structures. The audience navigates within the space, but does not have a physical presence, other than a navigational position within the virtual environment.

The audience has more of a presence inside the *The Singing Tree* simulation, created by Oliver *et al.* In *The Singing Tree*, the audience interacts with virtual feedback systems that are designed to make them active participants and 'performers' in the visualization experience. Similarly, visitors to Ikam and Fléri's *Elle et la voix* [4] installation take an active role in the simulation, as they modify *Elle*'s vocalizations by using their own voices.

Live theatrical performance and virtual imagery are combined in Levin and Lieberman's *Messa di Voce* performance piece. The audience is passive, as in a traditional theatrical setting, but observes a performance that simulates an integra-

tion between virtual imagery and the physical world upon the augmented theatrical stage.

Audience Interaction in Our Immersive Visualizations

The visualization metaphors presented in this thesis also combine the real world and the virtual world in ways that involve different types of audience interactivity.

The virtual ANIMUS character visualization described in Chapter 5 of this thesis is designed as a performance piece which displays a three-dimensional responsive character that is similar in size and scale to a physical performer. Similar to Levin and Lieberman's use of virtual imagery in theatre, this simulation facilitates the audience's illusion that a physical performer and a virtual character are interacting within the same space. The *Deep Surrender* responsive video project described in Chapter 6 of this thesis also exploits large-scale visual imagery that is used to augment the performance of a live vocalist who is observed by a viewing audience.

The Virtools simulation described in Chapter 7 of this thesis allows participants to navigate through an immersive virtual environment. This is similar to Ox's use of virtual reality space in the *Color Organ*. The audience in this simulation can also interact with the environment through vocalization. This is similar to the interactivity provided by Oliver *et al.* in the *Singing Tree* and by Ikam and Fléri in *Elle et la voix*.

3.3 Summary

In this Chapter, a survey of related research has been provided which provides an overview of techniques used to parameterize musical input and visualize it through responsive imagery.

- Signal processing systems used to extract musical feature data from acoustic input have been described, as have techniques used for determining melody

for the purposes of score-following. Similarities and differences between our musical feature extraction system and these signal processing systems have been highlighted.

- Visualizations which illustrate musical features through the behaviours of virtual characters have been described and compared to our own musically responsive ANIMUS characters.

- Several artistic installations which visualize music in immersive settings have been surveyed. Their influence upon our immersive visualizations has been noted.

Chapter 4

Extracting Features from Live Music

In order to extract meaningful information from a live musical performance, the stream of incoming live music must be parsed and organized in a way that facilitates further musical analysis. This chapter [1] describes the method by which we extract musical information and organize it within a schema that is consistent with Western tonal music theory.

Our musical feature data extraction module extracts the following parameters and transmits them to the visualization engines:

- Pitch – Vocal pitch is communicated in two ways (the raw pitch, and the pitch described in terms of its scale degree relative to the tonic of the key signature).
- Loudness – Vocal amplitude is transmitted in dB.
- Timbre – The singer's vowel choice is described. Open vowels (like /a:/) are differentiated from closed vowels (like /i:/).
- Chord – The chord the user is playing on the digital piano is identified.

We use these parameters to interact with our three music visualization applications.

[1] A version of this chapter has been published. Taylor, Boulanger and Torres, 2005. Proceedings of the 5th International Symposium on Smart Graphics. Springer LNCS, pages 13-24.

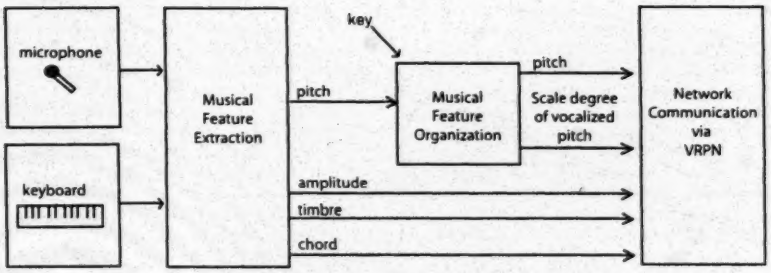


Figure 4.1: Musical Perception Filter Layer

- ANIMUS – The ANIMUS character is reactive to the emotive content conveyed by the tonal context (the scale degree) of sung pitches.
- Jitter – Responsive video is controlled by a singer's vocal timbre and the chords played on a digital piano.
- Virtools – Raw pitch and loudness data is used to generate particle clouds of various colours and sizes.

4.1 Overview of the Musical Perception Filter Layer

We have created a specialized perception layer called the Musical Perception Filter Layer (see Figure 4.1). The live musician interfaces with this layer by singing into a microphone and playing a digital piano. We operate the Musical Perception Filter Layer as a server, allowing us to easily use it as a real-time interaction mechanism for multiple visualization metaphors.

Our tasks within the Musical Perception Filter Layer are twofold. First we must parse a complex stream of incoming musical data in order to extract meaningful features upon which further analysis may be made. Second, we must consider how we will be using the musical data to control visualization in order to choose an organizational scheme for the extracted data that best serves our needs.

4.2 Extracting Musical Feature Data from Real-Time Audio and MIDI Signals

We use a Macintosh G5 system to extract important features from a stream of live musical input, and then communicate these extracted features across a network to various visual 'front ends'. Figure 1.1 shows a system diagram depicting how the Musical Perception Filter Layer interfaces with visualization engines.

4.2.1 Cycling '74's Max/MSP Environment

In order to parse the stream of live music, our system uses functionalities provided by Cycling '74's Max/MSP [7] development environment. Max/MSP provides users with a graphical environment within which they may create audio applications. Max is the fundamental development environment, handling basic dataflow operations and MIDI interactions while MSP is an add-on application layer comprising analogue audio analysis and synthesis routines. Together, they form a powerful toolset ideal for rapid development of audio processing applications.

Tod Winkler describes Max/MSP as a 'Fourth-Generation Language' [39], meaning an interactive development environment in which users can create fully-functional applications while avoiding low-level coding. The Max/MSP development environment approximates an object-oriented development strategy, allowing users to build applications by combining a series of *objects* (Max/MSP objects are analogous to functions in that they have inputs and outputs and can perform operations on data) connected by *patch cords* (the means by which data flows between objects). Max/MSP *patches* contain a sequence of objects connected by patch cords in order to perform a programming task. Winkler notes that the vernacular of the Max/MSP environment is loosely derived from that of the analogue music synthesis community, therefore the use of the *patch cord* metaphor and the visualization of dataflow is intended to be intuitive to their target user community. Max/MSP

patches are created visually, and upon execution operate by calling the underlying C-code corresponding to the flow of data described by the arrangement of objects and patch cords.

There exists a large community of Max/MSP users, many of whom have no formal training in computer programming or software development. Max/MSP's intuitive graphical user interface and object-oriented characteristics allows users to develop complex applications in small hierarchically organized pieces. Objects are connected to form patches which may then be used as sub-patches inside other patches. Max/MSP developers often share complex patches with other community members, facilitating community growth through collaboration.

A very useful feature of the Max/MSP development environment is the ability to code customized *external objects* in C which can then be used as part of the Max/MSP dataflow. Coding in C is of course, often faster and more efficient for experienced programmers. We make use of this functionality in creating our Musical Perception Filter Layer.

4.2.2 The Musical Perception Filter Layer Implemented as a Max/MSP Patch

The Musical Perception Filter Layer patch (see Figure 4.2) combines existing user-created Max/MSP objects with our own custom made objects in order to handle the task of extracting features from live musical input and communicating them across the network.

As shown in Figure 4.2, our patch contains three submodules:

- The Pitch Tracking Module (which encapsulates the `fiddle~` object [28]) extracts information about a singer's pitch, loudness and timbre.
- The Chord Detector Module identifies the chords played on the digital keyboard.

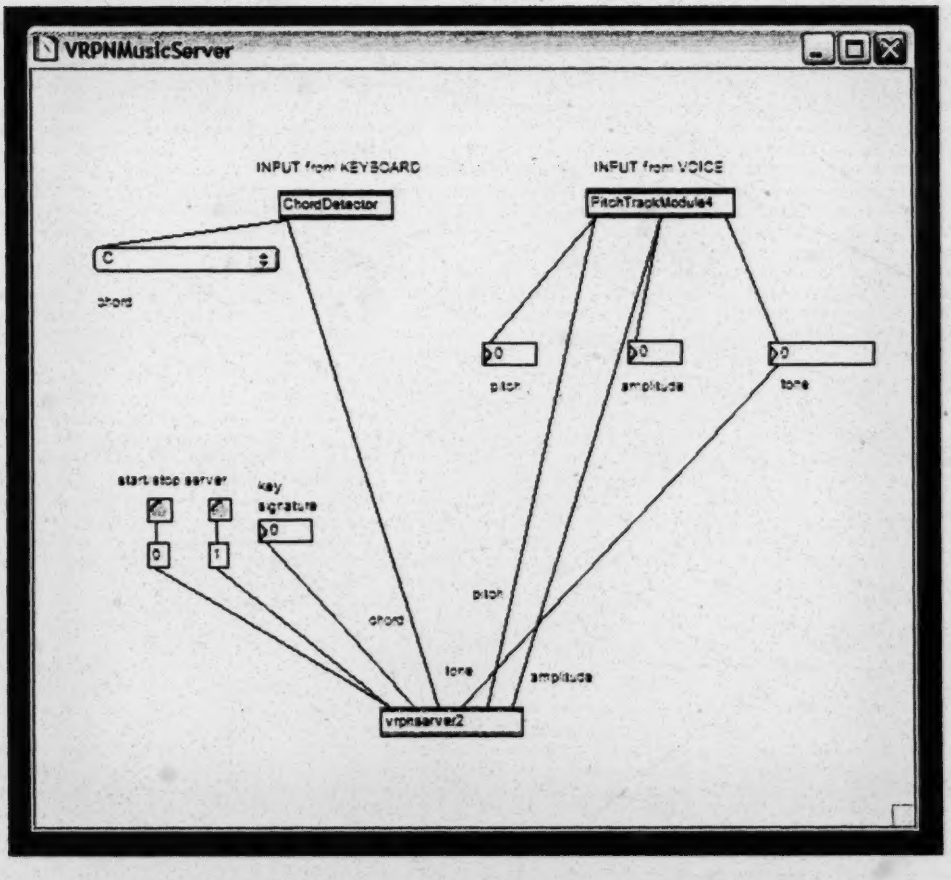


Figure 4.2: Musical Perception Filter Layer Max/MSP Patch

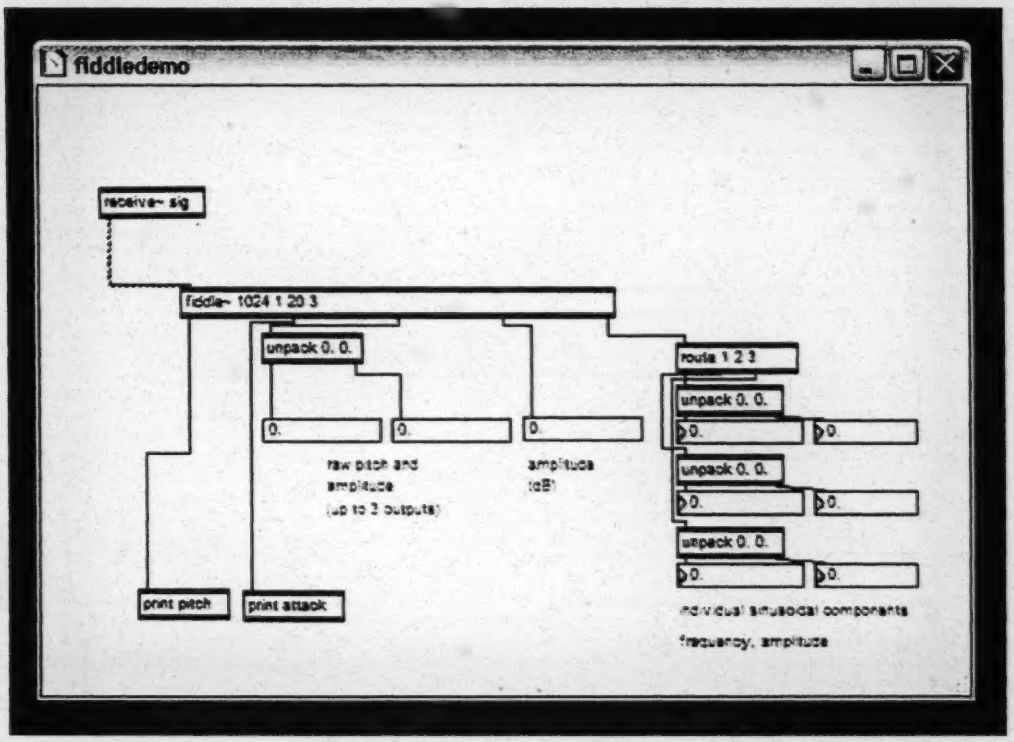


Figure 4.3: The `fiddle~` Object (as seen in the `fiddle~` help file) [28]

- The `vrpnserver` sends extracted data to connected visualization clients.

4.2.3 Extracting Vocal Information Using the `fiddle~` Object

Pitch, amplitude and timbral information are extracted from sung vocal input using the `fiddle~` object (see Figure 4.3) created by Puckette *et al.*[28].

Pitch and Amplitude Extraction

The `fiddle~` object analyzes the incoming sound signal to determine the pitch of the singer's vocalization. In order to identify the harmonic frequencies contained in the incoming signal, Fourier transformation is used to convert the signal's complex waveform into a harmonic spectrum. This spectrum contains data describing the frequency and peak amplitude of each sinusoidal component contained in the

incoming sound (see Chapter 2).

In order to determine which frequency is the fundamental frequency of the analog signal, Puckette *et al.* use a scheme they call a "likelihood function."

Their likelihood function is used to evaluate the likelihood that some suggested frequency, **f** is the fundamental frequency of the incoming sound signal. Each peak in the sound spectrum is assessed, and the likelihood of **f** being the fundamental frequency increases based on the peak's amplitude, how closely its frequency resembles a multiple of **f**, and whether its frequency resembles a low or high multiple of **f**. This likelihood function is evaluated for multiple values of **f**, and the frequency **f** with the highest likelihood is determined to be the fundamental frequency, or *pitch* of the sound.

The *amplitude* reported for each interval is estimated to equal the greatest amplitude found at any of the peaks.

The `fiddle~` object continually outputs these values, estimating the singer's pitch and amplitude.

Pitch extraction from acoustic input is not an easy task. How humans interpret musical sound is not necessarily how computerized pitch trackers will interpret it. One aspect of acoustic musical input which can cause pitch trackers to report incorrect pitch data is the fact that the fundamental frequency of acoustic musical sound may vary slightly during the different stages (attack, decay, sustain and release) encompassed by the sound's temporal envelope (see Chapter 2 for details.) Human perception adjusts for these small fluctuations and interprets the sound as having a consistent pitch rather than a stream of slightly differing pitches, but pitch tracking software may assess each temporal stage of the incoming sound, and report extraneous pitch data that humans would perceive as incorrect.

It is not that the pitch tracker is incorrect – rather, it is *too* correct, reporting small pitch fluctuations that we as human listeners do not consider relevant. Some-

times the `fiddle~` pitch tracker reports these extraneous pitches, therefore, the pitch data we report in the Musical Perception Filter Layer may contain a small amount of perceptually erroneous information.

Timbral Information Extraction

In addition to pitch and amplitude information, the `fiddle~` object also provides the frequency and amplitude data describing each of the partials forming the harmonic spectrum of the user's singing voice. We analyze this harmonic spectrum to obtain information about the singer's timbre.

Upon examination of this harmonic spectra, we assess whether the user's vocal tone amplitude is mainly concentrated at the fundamental frequency, or whether there is also significant tone amplitude distributed amongst the second and third partials forming the harmonic spectrum.

If we compare the harmonic spectrum that describes the singer's vocalization to known information about the harmonic spectra characterizing vowel production (see Chapter 2), we can describe an aspect of the singer's vocal timbre by providing a rough estimation of the user's vowel choice. This analysis could be further expanded in the future in order to produce a more detailed measure of vocal timbre, but currently produces a simple parameter that a vocalist can control by modifying the vocal tone he or she employs.

4.2.4 Monitoring Keyboard Input

Keyboard input monitoring is done in a separate sub-patch (see Figure 4.4.) Our Max/MSP sub-patch monitors MIDI events in order to determine what chords are being played on the keyboard. To do this, we determine which pitches are being played on the keyboard, and identify them in terms of *pitch class*. We consider the note **C** to have a pitch class of 0, **Db** to have a pitch class of 1, and so on. We examine the pitch classes being played on the keyboard, and identify the 'smallest'

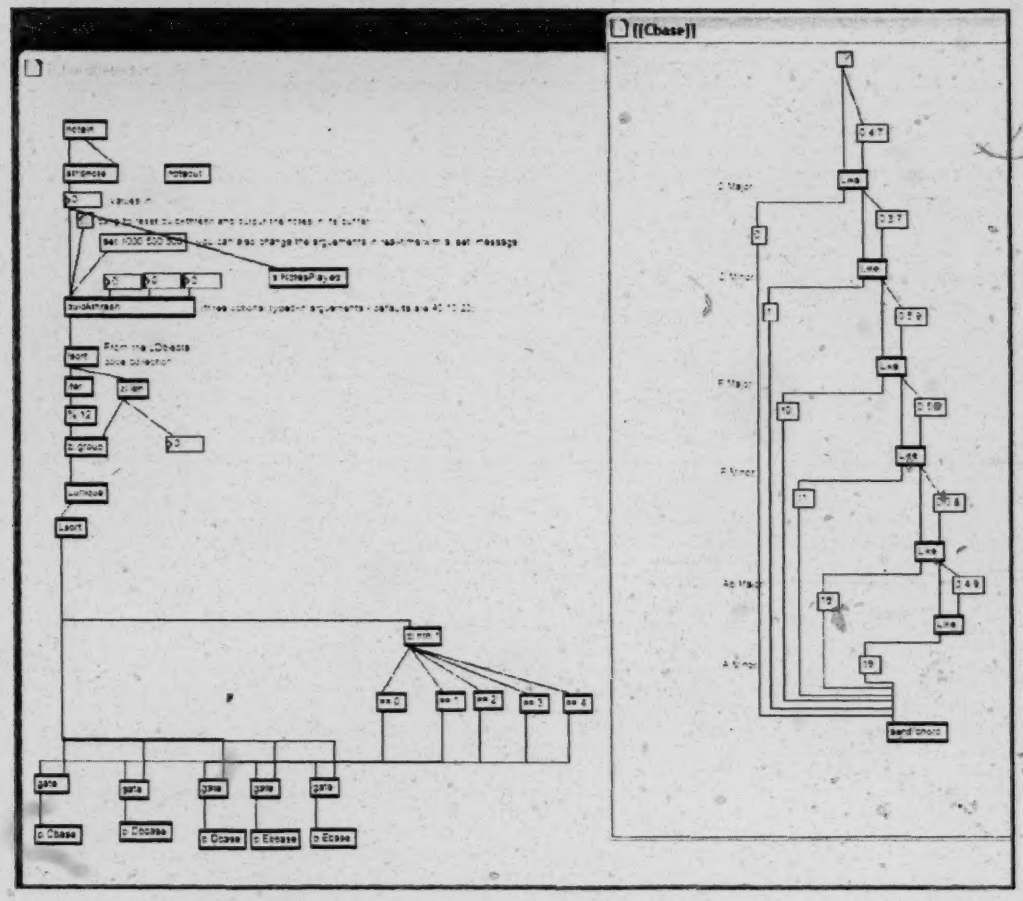


Figure 4.4: Chord Detector Sub-Patch

pitch class found (**C** with a pitch class of 0 would be considered 'smaller' than **E** with a pitch class of 4). We have formulated a list of 'known' major and minor chords, and classified them in terms of their smallest pitch class.

> **Example:** Consider the C minor chord (**C-Eb-G**). The smallest pitch class in that chord is 0 (the chord contains an instance of a **C**.) We look at the list of 'known' chords that have a smallest pitch class of 0 (see Figure 4.4) and find that the chord could possibly be one of several chords: C major, C minor, F major, F minor, Ab major or A minor. We compare the remaining pitch classes in the chord (**Eb** has a pitch class of 3 and **G** has a pitch class of 7) to this list of known possibilities, and determine that the chord is a C minor chord.

This method permits chords to be played in any inversion. More chord types (diminished, augmented, major and minor sevenths, etc.) could easily be classified and added to the list of known chords, making expansion of this module trivial.

4.3 A Tonal Encoding Model for Musical Feature Organization

We use Western tonal music as input melodies to our music visualization systems, so once our raw pitch data is extracted, we then organize it in a way consistent with tonal music theory.

4.3.1 Western Tonal Music

The harmonic structure used in most modern popular and folk music is Western tonality. Western tonality was also used extensively in pre-twentieth century classical music. It is a musical structure which is very familiar to North American and European audiences.

Western tonal music, while at the lowest level consisting of pitches, durations and amplitudes, is constrained by rules of harmonic structure. Music has a definite key signature, and melodies convey a sense of tension by moving to and from different tones in the key signature [5]. This tension is predictably resolved by a return to the tonic. Those who have not studied Western tonal music may not know how to describe what they 'expect' to hear, but the prevalence of this tonality scheme in popular musical culture makes its rules and patterns familiar to even casual listeners.

One of our visualization applications simulates human-like emotional responses to musical input. Organizing musical features within a tonal context (based on a schema devised by Deutsch and Feroe [9]) is consistent with previous research in the area of human musical perception indicating that humans exposed to Western tonal music internalize at some level its harmonic structure.

The brain activity of musically trained individuals shows a pronounced response when of the tonal rules of musical structure are disregarded. Patel *et al.*[25] report that a trained musician's brain activity responds to harmonic incongruity in the same way as it responds to linguistic incongruity, suggesting similarities between linguistic and harmonic syntactic processing. Physical response to tonal incongruity is not limited to musically trained individuals, however. Experiments in psychoacoustics by Koelsch *et al.* [15] have shown that perceivable deviations from harmonic structural rules cause specific patterns of brain activity in the brains of non-musicians who have been casually exposed to Western tonal music throughout their lives.

Additionally, encoding melody in terms of tonal context facilitates further music-theoretical analysis.

4.3.2 A Music-Theoretical Melody Encoding System

To organize vocal input within a tonal context, our system incorporates aspects of an existing music-theoretical melody encoding system devised by Deutsch and Feroe [9]. Their encoding system must, of course, be adapted for our needs.

Deutsch and Feroe describe the representation of absolute pitch values (which we extract from vocalization and keyboard input using Max/MSP patches) as the lowest level of musical information representation. They propose a higher-level system which integrates the raw pitch information into the tonal context within which it is contained. Their system assesses a musical phrase to identify an appropriate pitch *alphabet* (chromatic scale, diatonic scale, etc...) which contains each element of the phrase to be described. They choose a dominant event in the phrase to use as a *reference element*. Each note in the phrase is then described in terms of its position in the specified alphabet with relation to the reference element. Additionally, their model allows complex or repetitive phrases to be described in a hierarchical fashion.

We have complied with their notion of an alphabet, a reference element, and the specification of all other elements in terms of their relationship to the reference element. Our system does not encompass all the features of Deutsch and Feroe's model, as it is merely a subset of their extensive music theoretical model. Instead, it takes a more basic approach with the intent that it could be expanded in the future.

> **Example:** Within the key of C, the tonic note, **C**, would be the *reference note*. If the chosen *alphabet* was the chromatic scale (a scale which contains 12 semitones), **E** (a major third above C) would be represented as being *four steps in the alphabet above the tonic reference note* (each step would equal one semitone, since the alphabet is

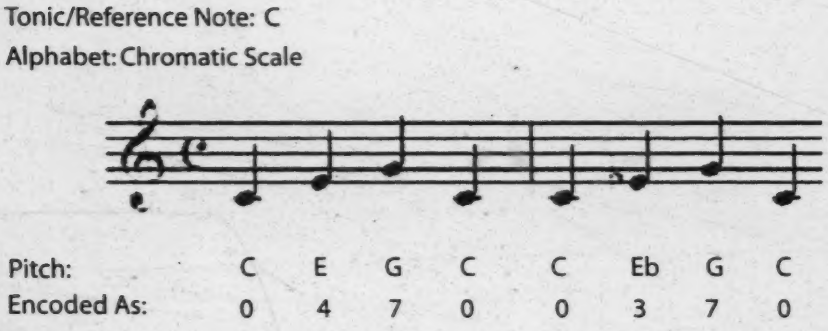



Figure 4.5: An example of pitch encoding

the chromatic scale) while **Eb** (a minor third above C) would be *three steps in the alphabet above the tonic reference note*. **G** (a perfect fifth above C) would be represented as being *seven steps in the alphabet above the tonic reference note* (see Figure 4.5.)

The real-time nature of our system makes our encoding task different from one based on a traditional harmonic analysis. We do not have the ability to assess an entire musical work at once, but rather must operate in a linear fashion as the musical input arrives. Our approach must therefore necessarily deviate from Deutsch and Feroe's in that, while their assessment of a dominant element in a melodic phrase benefits from the ability to make this determination after-the-fact, our system must address and encode melodic information as it is performed live, with no previous knowledge of the score. For this reason, we have chosen to use the tonic note of the key signature of the piece as the reference note. All further notes perceived are encoded as they are related to the tonic note of the key signature.

The melody encoding sub-module is implemented using C code inside the user-created `vrpnserver` module, so that pitches can be related to their tonal context before being transmitted to the visualization engines. We input the key signature used for encoding pitch via a parameter that the user can modify in Max/MSP.

4.4 System Architecture

This section provides a description of the data flow amongst components of the Musical Perception Filter Layer and between the Musical Perception Filter Layer and the visualization environments. Refer to Figure 4.6 for a detailed system diagram.

4.4.1 Musical Input

Musical input to the system comes from the vocal microphone and digital piano connected to the Macintosh system. Input from these devices is received by the Musical Perception Filter Layer, implemented in Max/MSP.

4.4.2 Musical Feature Extraction

Musical Feature Extraction is the first task handled inside the Musical Perception Filter Layer.

Feature extraction is implemented using two submodules:

- the vocal extraction module (described in Section 4.2.3 of this Chapter) uses the `fiddle~` object to extract information about sung pitch, amplitude, and timbre
- the chord detector module (described in Section 4.2.4 of this Chapter) identifies chords played on the digital keyboard

These extracted musical features can be communicated to the Responsive Video Environment using the data flow system shared by Max/MSP and Jitter. Communication with the Jitter environment does not require networking functionality, as Jitter is part of the Max/MSP development environment and runs on the same Macintosh PC as the Musical Perception Filter Layer. Chapter 6 of this thesis describes how musical data is visualized in the Responsive Video Environment.

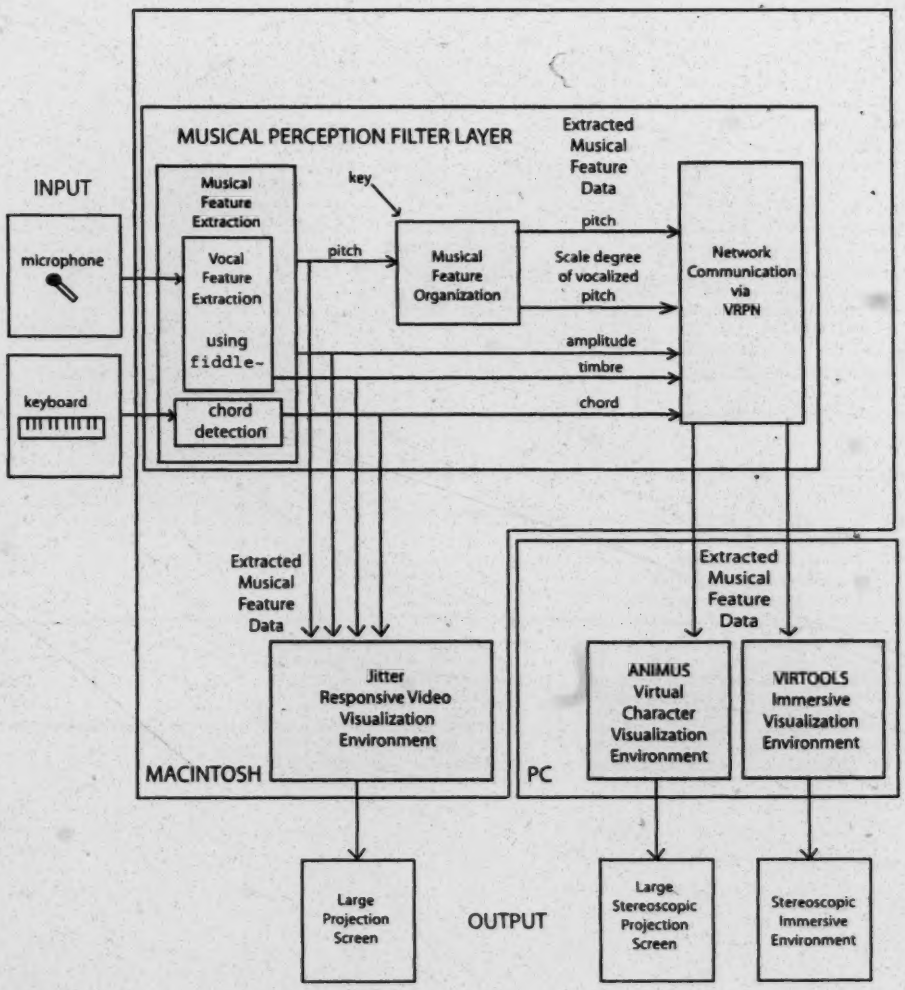


Figure 4.6: System Architecture

4.4.3 Musical Feature Organization

The second task handled by the Musical Perception Filter Layer is the task of pitch encoding.

Extracted pitch data destined for the ANIMUS and Virtools visualization environments is passed to the musical feature organization submodule described in Section 4.3 of this thesis. In this module, pitch information is organized within its tonal context before being transmitted across the network.

4.4.4 Network Communication using VRPN

The third task handled inside the Musical Perception Filter Layer is the task of communicating the musical feature data to the visualization environments.

After we have extracted and organized all the musical parameters (pitch, amplitude, vocal timbre, and chord data) we must then communicate them to the visualization engine via a network connection. This is done using the Virtual Reality Peripheral Network (VRPN) [35].

The VRPN Networking Package

VRPN is a client/server networking package designed specifically for use in virtual reality applications. VRPN server and client code provides a transparent wrapper hiding the details of socket communications. It is easy to implement networking functionality using the VRPN library. All a programmer must do is include the `vrpn.lib` file in a C++ project, and make a series of simple function calls in order to host a VRPN server or connect to a VRPN server as a VRPN client.

It is beneficial to use the VRPN library to facilitate network communication, as VRPN is highly portable. VRPN clients can be run in Windows, Macintosh, IRIX and Linux environments, making it possible to interact with visualization engines on any of those platforms. It is also integrated in commercial virtual reality devel-

opment environments such as Virtools [8]. The Max/MSP `vrpnserver` object is therefore very re-usable, as the next chapters will illustrate.

Integration of VRPN into Max/MSP

We have integrated VRPN into the Max environment by custom-creating our own Max external object, `vrpnserver`. Custom-created Max objects are coded in CodeWarrior, using a small wrapper class provided by Cycling '74 in order to define the object's inputs, outputs, and functionality.

Our `vrpnserver` object takes the extracted musical features as input, and calls functions from the VRPN library in order to run a VRPN server on the Macintosh that broadcasts the extracted musical feature data (pitch, loudness, timbre and chord data) to the network. A machine running a visual simulation can connect to `vrpnserver` as a client in order to access the extracted musical feature data.

Using VRPN to handle client/server socket communications, text strings are sent by the `vrpnserver` to the visualization clients in order to communicate musical feature data. Each time a musician sings or plays on the keyboard, `vrpnserver` sends a simple text string indicating the parameter being transmitted and the parameter's value. Sending a vocal pitch value equivalent to MIDI pitch 70 would be done by transmitting the text string "`P=70`". This string is parsed on the client computer running VRPN client code.

4.4.5 The Visualization Engines

The client computers running the ANIMUS Virtual Character simulation and the Virtools Immersive Environment simulation receive extracted musical feature data through their VRPN client modules.

Chapter 5 describes how the ANIMUS responsive character visualization illustrates the emotive capacity of a musical performance through interaction with a virtual character.

Chapter 7 describes how an application created in the Virtools environment visualizes incoming musical feature data in an immersive responsive environment.

The Jitter responsive video visualization receives musical feature data directly from the Max/MSP data flow. Chapter 6 describes how the responsive visualization application implemented in Jitter illustrates a singer's vocal performance.

4.5 Summary of Contributions

In this Chapter we have described the Musical Perception Filter Layer, which we will refer to throughout the rest of this thesis. The Musical Perception Filter Layer is responsible for extracting musical feature data, encoding melodic information within a tonal context, then transmitting the musical feature data to the visualization engines that associate musical content with responsive imagery.

Our Musical Perception Filter Layer has the following characteristics:

- The perceptual features of pitch, amplitude and timbre are extracted from live vocal input
- Chord data is identified from live keyboard input
- Pitch information is classified within its tonal context relative to a melody's key signature
- The Virtual Reality Peripheral Network library is used to create a musical feature data server that visualization engines can connect to in order to access the extracted feature data

Using the client/server module to separate the musical feature data extraction routines from the visualization engines facilitates code re-use when mapping musical feature data to various visual metaphors. Three examples of music visualizations implemented using the Musical Perception Filter Layer as a feature extraction system are presented in Chapters 5 through 7 of this thesis.

Chapter 5

Visualizing Music through Behaviourally Responsive ANIMUS Characters

This chapter [1] presents a method of visualizing live musical performance through the behavioural responses of a virtual character. The virtual character appears to hear the singing of a live performer, and responds in a way that makes it appear aware of the emotional mood the singer is expressing.

The virtual character is rendered on a large stereoscopic screen (see Figure 5.1.) The singer stands in front of the screen containing the life-sized rendering of the virtual character, making it appear that the character and the singer are both three-dimensional and equal in size relative to one another.

By using three-dimensional imagery that is comparable in size to the physical performer, we can blur the boundary between the real and virtual worlds, enabling the visualization to function as a physical-virtual environment that contains both the singer and the responsive character.

Torres and Boulanger's ANIMUS Project [37][38] enables the creation of animated characters that respond in real-time to the stimuli they encounter in the virtual world. ANIMUS characters are capable of perceiving events in their environ-

[1] A version of this chapter has been published. Taylor, Boulanger and Torres, 2005. Proceedings of the 5th International Symposium on Smart Graphics. Springer LNCS, pages 13-24.

Figure 5.1: The Musician Interacts with the Virtual Character, *Alebrije*

ment, formulating a cognitive response to these events, and expressing this response through dynamically generated animated behaviour.

I have collaborated with Torres and Boulanger to extend the ANIMUS framework (see [33] [34]) to allow music to be used as a way to interact with an ANIMUS character.

5.1 Chapter Overview

The work described in this chapter extends the ANIMUS framework to create a virtual character who appears to 'listen' to live musical input. The character can express simulated emotional responses through physical movements which visualize his mood.

This chapter will

- Provide an overview of Torres and Boulanger's ANIMUS Project
- Describe the relationship between the Musical Perception Filter Layer and the ANIMUS Perception Layer

- Present a method for simulating emotional cognition of music
- Discuss how responsive character movement is generated
- Illustrate our results by discussing an example of an ANIMUS character that responds to the emotive content of music

5.2 Torres and Boulanger's ANIMUS Project

Torres and Boulanger's ANIMUS Project [37] is a framework which facilitates the creation of 'believable' virtual characters. They describe a believable character as "[appearing] to be alive, giving the illusion of having its own thoughts, emotions, intention and personality." [38], p. 141.

ANIMUS characters respond to events in their environment in ways that illustrate their particular personalities. 'Shy' characters may cringe as if startled when another character makes a sudden movement, while 'curious' characters may come forward to investigate changes in their environment.

The ANIMUS Framework is designed with the intention to facilitate artist/scientist collaboration. Artists and scientists must work together and combine their talents in order to harness computer programming technologies to create synthetic characters that are capable of communicating believably with an audience of viewers. Artistic designers must be able to describe concepts to the technical teams that implement their visions. This requires that they be able to define their ideas in a clear and understandable way that corresponds to the programmers' implementation model.

To make this possible, Torres and Boulanger identify the need for highly modular system design. They break the task of information organization and processing into three distinct layers: the Perception Layer, the Cognition Layer and the Expression Layer.

The artistic designer can work with the developer to define virtual characters'

behaviour in terms of these three layers:

- What should the characters perceive in the world?
- How should the characters be affected by what they perceive?
- What visible feedback should be provided to the viewers in order to convey virtual character responses?

5.2.1 Perception Layer

Torres describes the ANIMUS Perception Layer as "an entry-point for the world, enabling the character to be aware of its environment within the bounds of its perceptual definitions." [36], p. 41. The ANIMUS Perception Layer represents the first processing stage in the ANIMUS Framework – it is the stage where information about the virtual world comes to the virtual character's attention.

ANIMUS characters must perceive features and events in the world around them in order to simulate believable response behaviour. Examples of perceivable events could include user input or actions of other characters in the virtual world. ANIMUS characters are programmed with specific perceptual capabilities. Some characters might be highly attuned to visible movement and notice even small motions made by any objects in their environment. Other characters might have very poor eyesight. They would not notice that an object was moving unless it came close enough so as to be within their small range of vision. Some characters might notice loud sounds, while others might be deaf.

For the purposes of implementing this layer, artists designing virtual characters must decide what their virtual characters need to perceive in the virtual environment. Developers can then implement the Perception Layer accordingly.

5.2.2 Cognition Layer

Torres defines the ANIMUS Cognition Layer as "the *spirit* inside the character." [36], p. 58. The ANIMUS Cognition Layer is where high-level cognitive tasks are simulated. Inside this layer, characters' particular preferences and dislikes are specified, their personalities are defined, and their simulated emotional states are determined.

Creating a cognitive system for a virtual character is by no means a simple task. However, it is important to note that the ANIMUS Cognition Layer does not need to replicate a realistic cognition system. The ANIMUS characters must be simply be *believable*, which is to say that they must appear to behave in ways that the viewing audience finds conceivable given the situation the characters are placed in and the input they perceive. "Like a human actor in a theatrical play" comments Torres, "[the ANIMUS character] convinces the audience of its authenticity." [36], p. 2.

Artistic designers must carefully define their character's cognitive capabilities in order for developers to implement an appropriate "personality" through a robust Cognition Layer.

5.2.3 Expression Layer

In the ANIMUS Expression Layer, Torres states that the virtual character must "stand before the audience and act!" [36], p. 67. The Expression Layer conveys the emotions, feelings and impulses generated in the Cognition Layer through visible animation. When an ANIMUS character has processed perceived data and determines that physical response behaviour is warranted, these behaviours are expressed through dynamically generated animation.

The ANIMUS Expression engine uses DirectX to generate character animations at run-time by using key-frame animation to interpolate between various combinations of pre-defined poses.

When designing ANIMUS characters, the artist must create a library of keyframe poses for each character using a 3D modelling program like Maya. Character designers choose keyframe poses from this library that can be combined to create animated behaviours. Designers can indicate how the generated animations should vary in mood and intensity in order to illustrate virtual characters' cognitive states.

5.2.4 The *Alebrije* Character

Torres created a virtual character in Maya, containing a poseable articulated skeleton. He named the character *Alebrije*, because as he explains, "Alebrije is a term used by some Mexican artisans to describe a particular kind of imaginary creature." [36], p. 38. Torres' lizard-like Alebrije character has a repertoire of poses from which animations may be generated, and was previously capable of perceiving the placements of objects in his environment. Alebrije would reflexively move his body away from objects in order to avoid collisions.

5.3 Creating Musically Responsive ANIMUS Characters

An ANIMUS character capable of expressing simulated emotional responses to the emotive content of live musical input was developed. The character processes live music and generates believable responses using the three-layered ANIMUS architecture:

- **Perception:** Using the Musical Perception Filter Layer to interface with the ANIMUS engine, an ANIMUS character is able to perceive musical input encoded in a way consistent with previously defined organizational models for tonal music.
- **Cognition:** The character is equipped with a cognition mechanism which relates perceived music-theoretical features to emotional states consistent with

the research of Cooke [5], who studied correlations between melody and emotional content. Using Cooke's metric as a way of mapping sound to emotion, the character is capable of simulating an emotional understanding of the music it 'hears.'

- **Expression:** The ANIMUS character expresses its simulated emotional response through visual animations which are generated in real-time and displayed on a life-sized stereoscopic screen. The character's expressive behavioural response animations provide a visual accompaniment to the performer's musical performance.

5.3.1 Interfacing with the Musical Perception Filter Layer

In order to create ANIMUS characters capable of simulating a believable emotive response to the content of a live musical performance, the characters must first be able to perceive important features in live musical performance.

Musically responsive ANIMUS characters must be able to 'listen' to the performance and perceive meaningful data from the incoming stream of sound. This is necessary in order for them to later assign cognitive meaning to aspects of the music they have 'heard.'

The specialized Musical Perception Filter Layer (previously described in Chapter 4 of this thesis) makes this possible. The Musical Perception Filter Layer extracts information from the live performance (pitch, amplitude, vocal timbre and chords played on the keyboard) and organizes it in a way consistent with Western tonal music theory. This information is transmitted to the ANIMUS Perception Layer so that it can be used as input to later cognition and expression processes.

ANIMUS characters receive and share information about the world around them using a 'blackboard' system. Information about events occurring within the virtual world is entered on the blackboard, and characters monitor this blackboard in order

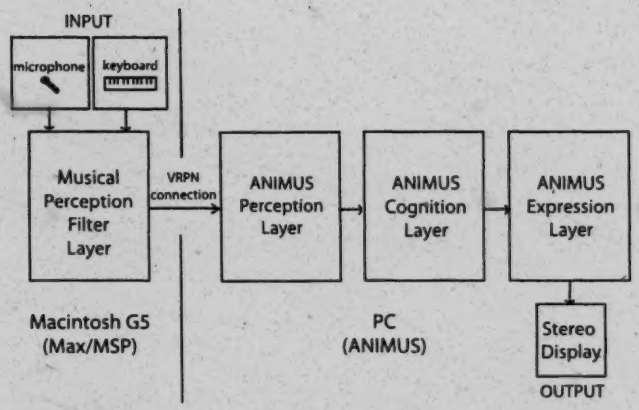


Figure 5.2: Integrating Musical Control into the ANIMUS Framework

to perceive these events.

To extract information from live musical performance, the ANIMUS Perception Layer connects as a client to the VRPN server running on the Macintosh, and obtains data (sung intervals, information about the singer's vocal timbre, and chord data describing what the user is playing) from the previously described Musical Perception Filter Layer (see Figure 5.2.) This data is then entered on the ANIMUS blackboard. ANIMUS characters monitor this blackboard to obtain information about the musical performance in order to carry out the cognitive and expressive tasks required to simulate responsive behaviour.

5.3.2 Simulating an Emotional Response in the Cognition Layer

In the cognition layer, the information sent by the perception layer must be received and analyzed in order to simulate the internal emotional state of the virtual character. The ANIMUS blackboard contains the musical feature data parsed from the live musical performance.

In order to formulate believable responses to live music, ANIMUS characters must simulate an emotional understanding of perceived musical features. The characters must analyze the input they have received in the Perception Layer and de-

termine how this input affects their simulated emotional state. The implementation of the Cognition Layer must address the question of how music should specifically affect the emotional state of each character, giving each character its own personal response to music's emotive capabilities. A cognitive awareness of the perceived data must be simulated in order for the ANIMUS character to have an interesting internal state which it can express through animated behaviours.

Cognitive awareness of musical performance could take many forms. Characters could assign feelings of happiness to a particular melody that they find pleasing, or they could enter a fearful state after perceiving a certain series of chords which they find threatening. Characters could dislike the piercing soprano of the Queen of the Night's coloratura, or express admiration for the rich mezzo tones of Carmen's "Habañera."

In order to create an interesting and flexible cognitive layer for our ANIMUS character, we have chosen to implement aspects of Cooke's research as described in "The Language of Music" [5].

Deryck Cooke's *The Language of Music*

In his influential 1959 work, Deryck Cooke analyses tonal music in an attempt to understand how a musical piece conveys emotional content to a listening audience.

Cooke uses the term 'content' to represent the sense of exhilaration, despair, or bliss which a listener might report feeling after listening to a powerful piece of music. Although content cannot easily be extracted from the written score in the same way that pitch names, key signatures or harmonic structure may be, Cooke defends his belief that emotional content is inherent to music. He describes content as "not a technical part [of a piece of music], but something more elusive: the interpretation which we put upon the interaction of the technical elements." [5], p.199.

> "In other words, the 'content' is inseparable from the music, except as an emotional experience derived from listening to the music. If we use the word to signify 'the emotion contained in the music', we must keep clear in our minds that the emotion is contained, not as a necklace in a box, to be taken out and examined, but as an electric current in the wire: if we touch the wire we shall get a shock, but there is no way whatsoever of making contact with the current without making contact with the wire." [5], p. 199.

Cooke's work references his extensive research in Western tonal music, makes hypotheses concerning the relationships between musical structures and composers' emotional intent, and cites numerous examples from musical literature to support his conjectures. His study of a widespread assortment of classical works provides a strategy for determining a generalized relationship between music and emotion.

Instead of implementing a character which enjoys one specific melody and dislikes another, we are interested in creating a character which is flexible enough to simulate an emotional response to Cooke's "electric-current in the wire" by assessing the emotion conveyed by music-theoretical features within a melody line. Cooke has discerned certain features to be salient features within a large number of musical pieces. He theorizes that certain features used in Western tonal music represent particular emotional concepts in a relatively universal way. By applying his rules to sung vocal melodies, we can assign cognitive meaning to elements of the live musical performance.

Cooke identifies "the basic expressive functions of all twelve notes of our scale." [5], p.89. If a melody contains many instances of the minor third, Cooke's theory states that the proper interpretation of the passage would be "stoic acceptance" or "tragedy." [5], p.90. He bases this inference upon many cited examples of musical passages expressive of tragic emotion which contain the minor third, such as

Violetta's deathbed scene from Verdi's "La Traviata." Conversely, Cooke cites the American folk song "Polly-wolly-doodle" to exemplify how a major third often signifies "concord" or "joy" [5], p.90.

As described in Chapter 4, our Musical Perception Filter Layer encodes all sung melody notes into tonal context relative to the existing key signature. We know whether each incoming pitch is a minor third, a perfect fifth, etc. This facilitates the process of linking each sung pitch to Cooke's associated emotional context. The emotional impact of each pitch is then used to modify the ANIMUS character's internal state and trigger responsive behaviour.

Simulating Musical Cognition in the ANIMUS Framework

The ANIMUS system uses a 'driver system' to control character behaviour. Familiar to many computer game users due to its use in "The Sims" [10], a driver system operates on the principle that once the level of a specific character feature reaches a target maximum or minimum, behaviour is triggered. Sims fall asleep on their feet when their energy driver reaches its minimum, but get up out of their beds when their energy driver is at its maximum value. Similarly, our ANIMUS characters use driver systems to express their emotional state.

According to Cooke's theories, a minor third signifies tragedy, while a major third signifies joy. An ANIMUS character can be implemented with a 'happiness driver', the level of which is increased if the singer sings a melody which contains many instances of major thirds, and decreases if a minor third is sung.

As a live stream of music comes in, each note affects the happiness driver. Notes which are deemed to be sad (minor thirds, minor sixths, etc.) diminish the character's happiness, while happy notes (major thirds, etc.) make the character's happiness level increase. Held notes continue affecting the happiness driver for the entire duration of their existence, making prolonged notes significantly effective in

affecting happiness levels.

Other extracted musical features (chords, vocal timbre, etc.) can similarly be linked to the ANIMUS character drivers in order to influence character response. Minor and major chords could easily be linked to ANIMUS drivers in order to improve or worsen character mood. The vocal timbre of a soprano could be perceived as comforting to a character, while a bass or baritone may be perceived as threatening. This simply requires the addition of specific drivers for each of the mood ranges designers wish the character to transition through.

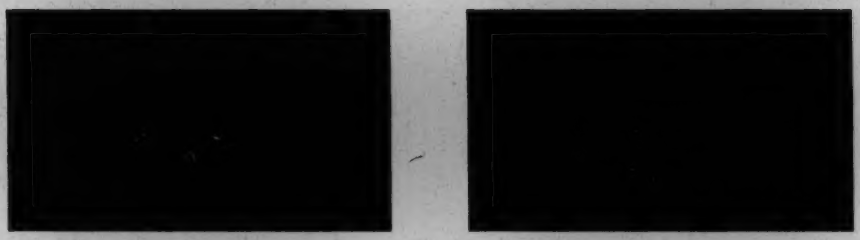
The cognitive processing of musical input allows the ANIMUS character to maintain a fluctuating emotional state during the course of a live musical performance. If the musician chooses to modify his/her melody spontaneously, the ANIMUS character's emotional state will adjust accordingly and appropriately. This emotional state is internal, however, and is not conveyed to the audience until it is processed by the ANIMUS expression layer, which uses animations to visualize character behaviour.

5.3.3 Visualizing Character Emotions Through Movement

An ANIMUS character is animated using keyframe-based interpolation. An ANIMUS character is a three-dimensional model that has a controllable skeleton. A variety of endpoint skeletal poses are defined using three-dimensional modelling software. All intermediate poses are generated at run-time in order to generate fluid transitions between these endpoint poses. This allows the animations to be dynamic, a key feature of the ANIMUS engine.

The ANIMUS expression engine allows the designer to define character animations both in terms of which keyframe poses are used to create a motion, and the speed of the transition between these specified poses.

These design decisions are linked to the ANIMUS character's cognitive state.



(a) The 'bored' pose (b) The 'attentive' pose

Figure 5.3: Alebrije in two poses

As a simple example, we describe an ANIMUS character who is attentive to vocalizations within his environment [34]. When the user of the system is silent, the character's cognition layer registers a high value in his 'boredom' driver. His expression layer translates this internal state by displaying him slumped in a 'bored' pose (see Figure 5.3(a)). When information about sung vocalizations reaches his perception layer, his cognition layer decreases its 'boredom' driver. His expression layer responds by generating a rapid transition from his 'bored' pose to his 'attentive' pose, adjusting the position of his head so that he looks towards the perceived source of the sound (see Figure 5.3(b)).

We can use the extracted emotive properties from a live musical performance as parameters which modify the character's actions. If we can infer an emotional narrative from a musical performance, extracted emotional indicators can be used to assist in selecting appropriate keyframe poses and transition rates when creating the character's animations.

5.4 An Example of a Musically Responsive Character

As an example of a musically responsive ANIMUS character, Torres' Alebrije character was extended to permit him to simulate an emotional response to major and

minor melodies.

We programmed Alebrije so that his simulated emotional state would become 'sad' if we presented him with a melody that contained many minor tones, like *Greensleeves*. His simulated mood improved if we sang him a piece containing major tones, like *Twinkle Twinkle Little Star*.

To give Alebrije the capability to react to the emotive significance of specific tonal tensions in the melody line, we outfitted his cognition layer with a 'sadness' driver, used to determine his level of unhappiness. He associated emotional meaning to the tonal context of major and minor pitches, consistent with the research of Cooke [5], and increased the value of his 'sadness' driver as minor pitches were perceived. Major pitches reduced the level of Alebrije's 'sadness' and transitioned him back to a neutral state.

Two poses were chosen to create Alebrije's expressive behaviour. He transitioned between a 'neutral' pose (see Figure 5.4(a)) and a 'sad' pose (see Figure 5.4(c).) The DirectX routines comprising Alebrije's expression layer were used at run-time to interpolate between the keyframe poses in order to generate intermediate poses (see Figure 5.4(b)) reflecting his current mood as described by his sadness driver.

5.4.1 Example: Greensleeves

To illustrate how Alebrije responded to minor melodies by simulating 'sad' behaviour, let us consider a visualization of a passage from the poignant folk song, *Greensleeves*.

We can examine the melody of the first two lines (see Figure 5.5). Minor melodies are usually transcribed in minor keys to reduce the number of accidentals (flats and sharps) used, but for simplicity's sake, the melody has been transcribed in the key of C major, using C as our tonic and reference note. We use accidentals

(a) The 'neutral' pose

(b) An intermediate pose

(c) The 'sad' pose

Figure 5.4: The Singer Interacting with Alebrije

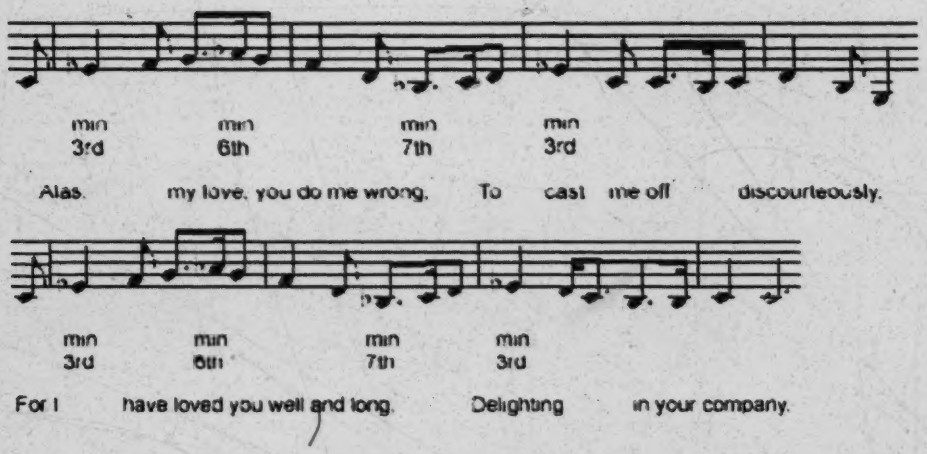

min 3rd
min 6th
min 7th
min 3rd
Alas, my love, you do me wrong, To cast me off discourteously.
min 3rd
min 6th
min 7th
min 3rd
For I have loved you well and long, Delighting in your company.

Figure 5.5: Greensleeves Melody

where necessary to denote notes such as Eb and Bb which lie outside of the C major key signature.

Alebrije's cognition layer was equipped with an rudimentary emotional understanding of the emotive function of major and minor tones. Alebrije was programmed to respond to minor tones (labelled in Figure 5.5 as the Minor Third, Minor Sixth and Minor Seventh) in a way that transitioned him towards his 'sad' pose. He was programmed to respond to major tones in a way that reduced his unhappiness and transitioned him towards his 'neutral' pose.

Alebrije began "listening" to the musical passage in the fully neutral pose. The result we desired from this implementation was that at the end of the musical passage, he would have accumulated a high enough value in his sadness driver that he would have fully transitioned to his unhappy pose. We found that we required some fine-tuning of the implementation as well as stylistic manipulation of the sung input in order to achieve an interesting and appropriate visualization.

5.4.2 Observations on Alebrije's Musical Responsivity

To test the system, I switched roles from 'developer' to 'vocalist', and attempted to interact with the Alebrije character by singing the Greensleeves melody and observing his responses.

The observations I made must be tempered by the fact that I am extremely familiar with the interface design and character programming; however, I made every effort to fine tune the system so that it did not require technical considerations from the artist.

Initially, I attempted to sing the Greensleeves melody while adhering to the timing denoted in Figure 5.5. This resulted in a disappointing animation. The minor tones appeared too briefly in the passage to have a significant effect on pose transitions. This resulted in an animation that did not capture the mood of the musical performance. Alebrije did not look sad.

Programmatic Revisions

When the visualization was first run, major and minor tones had equal impact upon Alebrije's sadness driver.

If the melody of Greensleeves is examined, most of the notes are in fact major, but the overall melody can still be characterized as minor. A small programmatic revision was introduced to address this. Minor tones were given a slightly greater impact on character positioning than the major tones. This was done by increasing the scale factor by which minor tones increased the value of the sadness driver.

Care had to be taken, that programmatically fine-tuning the implementation so that the Greensleeves melody would be well visualized did not make the implementation song-specific. For this reason, programmatic revisions were restricted to small re-weightings of the impact that major and minor tones had on the 'sadness' driver.

The programmatic revision improved the animation somewhat. Alebrjie made visible transitions towards sadness when the minor tones were sung, however the overall animation still appeared jerky and unnatural.

Singer/Character Interaction

The best results were achieved when I allowed myself to adjust my singing style in order to coax the best performance from Alebrije. Without further tuning the parameters correlating major and minor tones with pose transition rate, I instead modified my own performance and was able to control Alebrije's responses. To do this, I watched Alebrije carefully. When I saw him express 'sadness' in response to my vocalization, I modified the timing of the phrase slightly in order to emphasize (through extended duration) the effect of the 'sad' notes. This type of phrase manipulation is often intuitively used as a stylistic choice by jazz and folk singers, in order to allow them to emphasize important aspects of a melody through timing. The excess time given to stressed notes can be 'made up' by shortening non-significant notes in the phrase so that the singer does not fall inordinately out-of-time.

When I gave myself improvisational leeway, and a "goal" ("make Alebrije sad") I was able to elicit an interesting animation from Alebrije. Alebrije appeared to be cringing each time the melody lingered on a minor tone, and relaxing slightly when the melody returned to neutral tones. By the end of the passage, Alebrije's pose was quite definitely similar to his 'unhappy' position. The visualization was clearly responsive, as my lingering over the minor notes was visually matched by Alebrije's unhappy movements.

The observed interplay between my vocalization and Alebrije's movements was interesting. I was supposed to control Alebrije's movements, but by observing Alebrije's feedback, I found myself modifying my own behaviours in response. This was an unexpected interaction, but one which merits further exploration.

Pitch Tracking Errors

When testing this implementation, it was determined that the way we incorporated the `fiddle~` object's pitch tracking capabilities was not flexible enough to provide consistently good results in practice. The melodic analysis was working with discrete pitch estimates of vocal tone, rounded to semitone precision. Due to the fluctuation of fundamental frequency that sometimes occurs during different phases of a sound's temporal envelope (see the discussion of pitch tracking in Chapter 4 and the discussion of a sound's temporal envelope in Chapter 2) the `fiddle~` object's rounded pitch estimate was sometimes flat or sharp by a semitone, making the analysis of a note's tonal and emotive context incorrect. Implementing a score-following mechanism such as the one presented by Puckette *et al.* [29] into the system could help avoid this type of error. The ability to compare the incoming frequencies with a pre-defined score when making a pitch estimation would allow the system to make a better assessment of the singer's melody.

Another limitation of our system related to the nature of the pitch-tracking mechanism is that currently the singer must be very careful to begin her melody on the tonic of the specified key signature, so that each pitch in the melody can be correctly encoded into the tonal context of the pre-determined key. A better solution to this problem would see the adaptive tuning strategy described by McNab *et al.* [21] incorporated into the encoding process. If we encoded the melody by assuming that the first note of the piece was the tonic, with the rest of the notes encoded accordingly, this would remove the need for a singer to begin on a specific pitch.

5.4.3 Summary of Contributions

This chapter has described a music visualization application implemented using Torres and Boulanger's ANIMUS Framework [37] [38].

We theorized that by implementing a virtual character capable of ascribing emotional meaning to the tonal context of sung pitches in a way that was consistent with Cooke's research, we could illustrate the emotions contained in a musical passage. Our musically responsive ANIMUS character is capable of making transitions between designated poses in order to indicate the emotion conveyed by the major or minor tones in a melody.

Our visualization application makes the following contributions to musically responsive virtual character research:

- The extension of the ANIMUS Framework to handle musical input
- The creation of a character cognition mechanism capable of identifying emotion conveyed in melodies
- The visualization of compelling character response to live music through life-sized stereoscopic animation facilitated by the ANIMUS expression layer

We have explained that our melodic detection routine is not currently as accurate and flexible as we would like it to be. We have suggested two ways to improve the robustness of the melody tracking, including the implementation of a score-following algorithm (such as that described by Puckette and Lippe [29]) or a system of adaptive tuning (such as that described by McNab *et al.* [21].)

We would like to expand our virtual character's cognitive mechanisms and expression pose library in the future in order to increase the visualization's illustrative capabilities. A digital media artist is currently working with us to develop an expressive pose library for a virtual humanoid character that will be controllable using our musically responsive virtual character technology.

Chapter 6

Visualizing Music Using Responsive Video

This chapter describes the technologies used to create a performable multimedia work, *Deep Surrender*. *Deep Surrender* is a multimedia piece written for soprano, synthesizer, and responsive video. The intention of the piece is to illustrate the way an artist can harness anxiety and adrenalin to produce a beautiful performance. This illustration is done through the visual metaphor of a jellyfish – a creature both beautiful and terrifying. The artist's musical performance affects the jellyfish representation, in order to visualize how she can interact with and overcome her anxiety.

When creating this production, artistic and technical goals were formulated. The artistic goal for this piece was to create a visual metaphor capable of telling the story of the artist and her relationship with anxiety and adrenalin. The technical goal pursued in the development of this project was to create a responsive interaction platform that a performer could use to manipulate visual imagery in a fluid and natural way.

To allow a performer to control aspects of a responsive video visualization, data from live musical input is extracted using the Musical Perception Filter Layer described previously in this thesis (see Chapter 4). This extracted data is used to control video processing routines. Video manipulation is responsive to the musician's performance, allowing the media piece to be dynamic and expressive of the

Figure 6.1: *Deep Surrender* Live Performance

nuances of live performance.

The completed piece was performed in concert at the University of Alberta on December 14th, 2004 (see Figure 6.1.)

6.1 Visualization Mechanism

Video processing in this application is handled by Cycling '74's Jitter package [6]. Video processing Jitter objects (colour balance modification and chroma-keying) are used to generate the visual effects used in the piece.

6.1.1 Cycling '74's Jitter

Cycling '74's Jitter [6] is a Max/MSP [7] add-on that integrates image processing into the visual development environment. Jitter provides functionality for video and still image manipulation, as well as supports 2D and 3D graphics functionality using OpenGL. Jitter objects can be connected to Max/MSP objects using patch

cords, allowing them to fit natively inside the Max/MSP dataflow.

Jitter uses matrices to store all video and image data, and allows these matrices to be manipulated without limitation. Additionally, any other type of data (sound information, numerical data, etc.) may also be stored and processed inside a matrix, allowing users to experiment with the visual representation of non-image data. Numerous Jitter objects are provided which can perform arithmetic and logical operations upon matrix data.

Max/MSP music analysis can be used to data to regulate how Jitter's matrix operators modify video streams. This allows music to be used as an input system for responsive modification of video playback.

6.1.2 Selected Video Manipulation Techniques

The visual effects in *Deep Surrender* are done by layering different video streams upon one another and affecting their colour properties using Jitter functionality.

Colour Balance

Jitter allows users to easily adjust the colour balance of input video. Remembering that Jitter treats image data as a matrix, colour balance adjustments are done using basic matrix arithmetic.

Jitter stores imagery as a 4-dimensional matrix, with the matrix dimensions representing the red, green and blue channels as well as the alpha component signifying pixel transparency. In order to affect the playback colour of the video image, Jitter functionality allows users to scale the values in each of the colour dimensions.

To do this, a message is sent to the Jitter object `jit.scalebias` with an argument of `rscale`, `gscale` or `bscale` in order to identify the colour dimension to be scaled. Scaling a matrix's red dimension by a number greater than one will increase the redness of the image. Scaling by a number smaller than one will reduce the redness of the image.

This Jitter functionality is used to affect the colour balance of the video images in the *Deep Surrender* multimedia performance. Later in this chapter, details will be provided describing how colour balance is adjusted in response to chords input on the digital piano, and in response to the singer's vocal timbre.

When colour balance modification is combined with chroma-keyed video layering, more complex visual effects can be created. Specific elements in the video image can be modified, while others remain untouched.

Chroma-keying

The Jitter documentation describes Chroma-keying as 'the process of superimposing one image on top of another by selective replacement of color.' Chroma-keying is the technical term for 'bluescreening' or 'greenscreening', which is used in many films and television programs to create the illusion that actors are interacting with fantastical worlds or characters. In reality the actors are performing their actions against a blue or green backdrop which is then selectively replaced with a more interesting environment. Chroma-keying can be used to blend real actors and sets with computer-generated content, or to merge aspects of two different video streams into one composite video.

To chroma-key an image, a target colour is chosen as a reference colour, and then all pixels which contain colour values within a certain tolerance range of the reference colour are selected, or 'keyed'. These keyed pixels in the original image can then be filled with the colour values found in the corresponding pixels of an alternate image. This is why chroma-keyed videos are often shot against a solid blue or green backdrop – these unnatural colours are easy to select out of an image, since they do not naturally appear in human skin tones, eliminating the problem of incorrectly incorporating pixels which represent parts of the actors into the group of pixels to be replaced.

Figure 6.2: Original Image x Mask Matrix = Modified Original Image

Figure 6.3: Replacement Image x Mask Matrix = Modified Replacement Image

Jitter provides chroma-keying functionality through its `jit.chromakey` object. As jitter represents all imagery in matrix format, chroma-keying is done by creating mask matrices used to multiply the original and replacement image matrices.

A mask matrix is created which contains pixel values of zero in all the pixel positions which were keyed in the original image, and ones in the pixel positions corresponding to the un-keyed areas of the image (the area we wish to retain.) The original image matrix is then multiplied by this mask matrix. The modified original image then contains zero regions where the keyed pixels used to be (see Figure 6.2.)

The inverse of the keying mask is taken, resulting in a second mask matrix containing ones to represent the keyed portions of the original image, and zeros representing the un-keyed portion of the image (the portion intended for retention.) The replacement image is then multiplied by this mask matrix. This results in a modified replacement image that contains zeroed regions corresponding to the un-keyed regions of the original matrix (see Figure 6.3.)

The modified original image and the modified replacement image can then be

Figure 6.4: The Resulting Chroma-keyed Image

added together, creating a composite image matrix containing the retained portion of the original image with the replacement imagery superimposed into the keyed regions (see Figure 6.4.)

Chroma-keying can be used on moving video images in the same way as it is used on still pictures.

Combining Chroma-keying and Colour Manipulation

In the *Deep Surrender* project, chroma-keying and colour balance modification are used in tandem to create the visual effect that certain jellyfish within the environment are changing colour, while the environment remains unchanged.

This is done by compositing colour-adjusted video streams in a multi-step process:

- The video stream containing the original jellyfish image is duplicated into a second Jitter matrix.
- The colour balance of the duplicate stream is altered to the desired colour.
- The duplicated (colour-altered) jellyfish image is chroma-keyed into the original stream, overlaying the original jellyfish

The resulting video stream appears to contain a colour-manipulated jellyfish, while the rest of the image remains unchanged.

6.2 Visualized Parameters

In *Deep Surrender*, the Musical Perception Filter Layer is used to extract feature data describing the vocal timbre of a singer, and the chords played on a digital piano. In this visualization, chords are related to one another with regards to their positions on the music theoretical device, *The Circle of Fifths*.

6.2.1 Vocal Timbre

Johan Sundberg (author of *The Science of the Singing Voice*[32]) describes how a sung note, a note which has a certain pitch and a certain loudness, also has associated timbral properties: vocal colour and vowel quality.

A singing voice might colloquially be described as "dark", or "bright", "rich", or "thin". These would be examples of attempts to classify a singer's *vocal colour*. Vocal colour is determined by physiological characteristics of the vocal tract. These physiological characteristics can be manipulated by the individual's control over the vocal tract musculature.

Sundberg describes the vocal tract as a *resonator* which resonates most optimally at particular frequencies (known as *formant frequencies*.) How sound energy is distributed amongst these formant frequencies determines the timbral properties of a sung sound. He explains that "the shape and size of the pharynx and mouth cavities impose their characteristics on the sound of a person's voice" [32], p.2. by determining where the individual's formant frequencies reside.

By exerting muscular control over vocal tract, jaw, and tongue position, people can manipulate the placement of their formant frequencies. While untrained singers are generally unconscious of this vocal tract manipulation, this is how humans con-

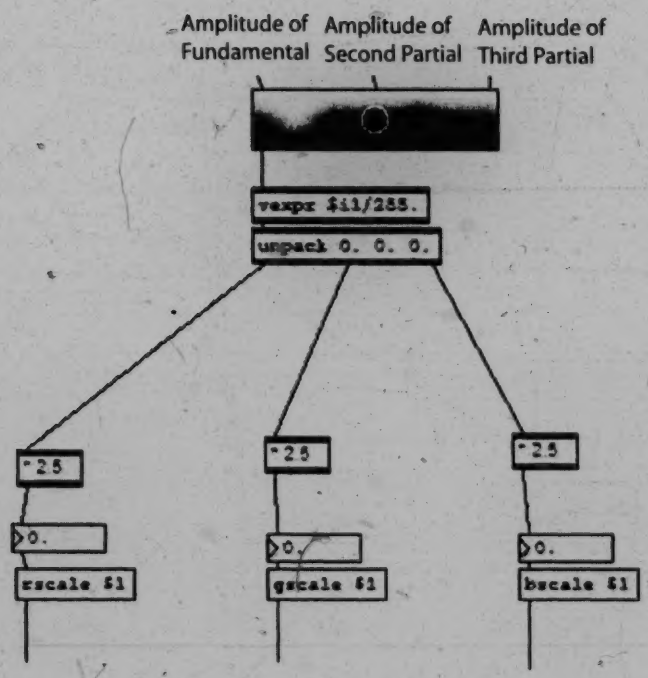


Figure 6.5: Visualizing Tone through Colour

trol vowel sounds in speech and singing, and how trained singers deliberately manipulate the subtleties of their vocal colour and vowel tone.

6.2.2 Visualizing Vocal Timbre

In the *Deep Surrender* visualization, vocal timbre is illustrated by visually representing the distribution of energy amongst the partials in the sound. This energy distribution is determined in the Musical Perception Filter Layer using the output of Puckette *et. al*'s `fiddle~` object [28]. In addition to outputting an estimate of

a singer's pitch and amplitude, the `fiddle~` object outputs raw data describing the fundamental and the other partials found in the sound. `fiddle~` outputs the frequency and amplitude of each of the sinusoidal components (the partials of the sound.)

As described in Chapter 2 of this thesis, a sung sound does not consist of simply one frequency. Instead it consists of a *fundamental frequency* and a series of *harmonics* or *partials* that exist at multiples of the fundamental frequency (the first partial exists at two times the fundamental, the second at three times the fundamental, and so on...) The distribution of energy amongst the the partials in the sound determine the vowel sound the singer is vocalizing. Examining the amplitudes of the partial frequencies in an audio stream can yield an estimate of vowel quality.

To map vowel sounds to colours in this visualization, the first three amplitudes (the amplitudes of the fundamental and the first two partial frequencies) are linked to the inputs of a Max colour-chooser object (see Figure 6.5) which takes as its input parameters red, green, and blue colour component values. This results in the generation of an RGB colour value that varies dependent upon the weighting of tone amplitude amongst the partials found in the singer's vocal output.

Assigning colour in this way yields predictable and repeatable results when a soprano voice (*Deep Surrender* is a soprano piece) is used as input.

Let us examine what happens if the soprano sings in the middle portion of her range. As an example, assume she is singing on the note **A4** which has the frequency of 440 Hz.

If the soprano sings the closed vowels /i:/ as in 'free' or /u:/ as in 'fool', tone amplitude is highest at the fundamental (see Figure 2.8) and the resulting colour output is in the red-to-yellow range. If she sings the open vowel /a:/ as in 'car', the colour output is in the green-blue range, as the amplitude of the second and third partials increases.

When the soprano approaches the higher extremes of her range (the soprano 'high C' at 1000 Hz and above), it is noted that vowel choice has less of an effect on tone amplitude concentration. At the extremes of the soprano range, tone amplitude is concentrated heavily at the fundamental, producing a consistently reddish colour output value.

Sundberg's research [32] provides us with evidence to support the validity of these results. He describes the first formant frequency of the /i:/ vowel to be approximately 300 Hz. In our first example, the soprano is singing a pitch of 440 Hz. Sundberg's studies of professional soprano vocal production show that when the sung pitch (the 440 Hz) is higher than the vowel characterizing formant, that the formant is modified so that is it is no lower than the sung pitch. This would account for the `fiddle~` object's describing /i:/ and /u:/ vowels as exhibiting high tone amplitude concentration at the first partial – the fundamental.

It is known that the first formant frequency characterizing the /a:/ vowel (see Figure 2.8) occurs in a higher vicinity (roughly 1000Hz) which accounts for the high weighting on the second partial in our above example (the second partial found for a sound with fundamental 400Hz would be found at 880Hz.)

When the soprano's pitch reaches her top register, Sundberg's observation that formants are modified so as to remain above sung pitch concurs with our observation – the modified formants appear in the region slightly above the fundamental frequency, accounting for the resulting red colour output observed for all vowel sounds at extremely high pitches.

In this visualization, a mapping between vowel type and colour is used to modify the jellyfish representations that are chroma-keyed into the video imagery. In this way, extreme sounds (the high pitched notes or closed vowels) can be characterized with the vibrant reds and oranges, and less dramatic sounds (the relaxed and open /a:/ vowel at moderate pitches) with the more restful blues and greens. This

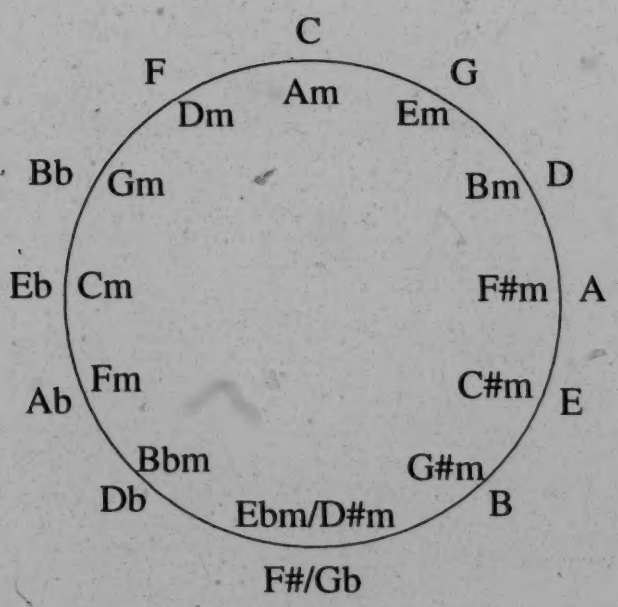


Figure 6.6: The Circle of Fifths

colour dynamic is used as a way to visualize the intensity of a singer's vocalization and to convey the artistic concept of the piece.

6.2.3 The Circle of Fifths

The Circle of Fifths (see Figure 6.6) is a music-theoretical device that represents the way the twelve major and twelve minor key signatures relate to one another [16]. Each key signature is rooted by one of the twelve notes of the chromatic scale. If the Circle is traversed in a clockwise manner, each subsequent step to the right on the Circle represents an interval of a Perfect Fifth. If it is traversed in a counter-clockwise manner, each step to the left represents an interval of a Perfect Fourth. These relationships (perfect fifths and perfect fourths) are significant, and appear frequently in tonal music. The I-IV-V chord progression (C-F-G would be an example of such a progression) appears often in popular music.

The placement of chords on the Circle of Fifths indicates the similarity between

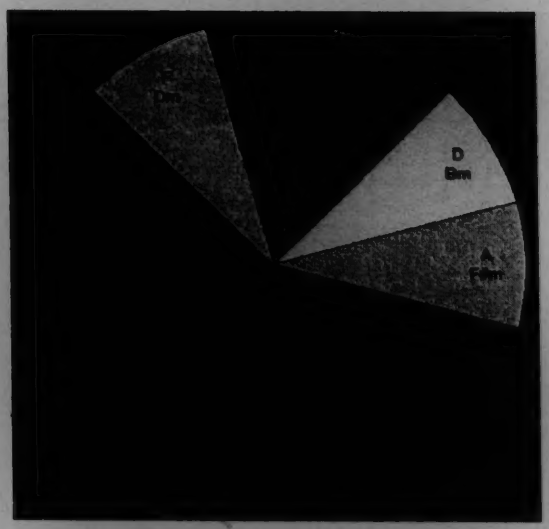


Figure 6.7: Mapping the Circle of Fifths to the Colour Wheel

the chords. C major and A minor appear in the same position on the Circle. This is because A minor is the *relative minor* of C major. Both C major and A minor are key signatures that contain no sharps or flats. E minor is the relative minor of G major (both E minor and G major contain one sharp) and so on.

The Circle of Fifths is a geometrical representation of the mathematical relationships structured into the tonal music system. As such, it is useful to study the placement of key signatures on the circle in order to visualize how they relate to one another.

6.2.4 Visualizing the Circle of Fifths

In this application the key signature relationships contained in the Circle of Fifths are visualized by mapping the Circle to the colour wheel (see Figure 6.7.) Chords that are adjacent on the Circle will have similar colour values.

The parallel between the circular structure of the Circle of Fifths and the colour wheel has been exploited in previous music visualization works. Jack Ox used a similar mapping in her *Color Organ* visualization [24]. Ox's work used the colour-

associated Circle of Fifths to visualize the existing chord relationships in musical pieces.

The mapping between the colour circle and the Circle of Fifths was used for a subtly different purpose during the compositional process of creating *Deep Surrender*. Since the mapping between chords and colours was fixed, the colours and chords could be treated synonymously. In order to express the emotional intentions in the piece, the song was composed visually, by choosing the appropriate colours on the wheel to convey emotions. The sound of the song was determined by choosing between the major and minor chords found at the selected colour location. This ensured that the visual colour scheme in *Deep Surrender* accurately reflected the emotional intentions behind the composition, as the colour choices were the most influential parameter considered in the creative process of the piece.

6.3 The *Deep Surrender* Multimedia Project

6.3.1 Part One

In Part One, the performer is merely an observer of the fearful environment. The visualization is simple, with the artist controlling only the colour of the image by playing simple chord progressions.

- The bold colours in this section are controlled through the `jit.scalebias` object which is triggered by keyboard input.

In Figure 6.8 we see the results of the performer playing a G major chord on the keyboard, which corresponds to the orange-yellow colour tone.

6.3.2 Part Two

In Part Two, the performer begins to experiment. She uses her voice to introduce entities to the visual environment, and realizes that her actions elicit a colourful response from the ominous creatures.

Figure 6.8: Deep Surrender Part One

- The composition modulates to an eerie set of chords which correspond to icy colour tones in the colour spectrum (purples, pinks, blues)
- A dark and sinister video sequence begins
- The colour balance is still modified by keyboard input
- The performer's voice brings new and vibrantly coloured entities into the environment, using the `jit.chromakey` functionality
- The performer's voice is used to offset the composed colour choices, making this section visually dynamic

In Figure 6.9 we see the orange jellyfish introduced to the scene as a result of the performer's vocalization. The jellyfish is orange because she is using a focused tone on a closed vowel (/i:/) which corresponds to a red-orange colour.

6.3.3 Part Three

In Part Three, the performer realizes that the most beautiful results are produced when she combines her voice with the fearful imagery. The visual effects flourish

Figure 6.9: Deep Surrender Part Two

Figure 6.10: Deep Surrender Part Three

through her willingness to overcome her hesitation and thrive upon the adrenalin rush of fear.

- The song returns to the original theme, and the first video is replayed
- Additional sound layers join the synthesized accompaniment, helping to convey new energy in this section
- Using her voice as input, the performer can modify the video playback (through chroma-keying applied to duplicate image streams) to produce the piece's most vivid colour effects

In Figure 6.10 the performer uses the /i:/ vowel to superimpose the red tones upon the scene. The underlying colour balance is controlled by the keyboard (the green colour is produced by the A chord) and the vocalizations affect the highlighted portions of the jellyfish, giving an intense visual result.

6.3.4 Credits

Deep Surrender: A Piece for Soprano, Synthesizer, and Responsive Video

Composition and Performance – Robyn Taylor

Audio/Video Processing – Robyn Taylor

Source Video – Melanie Gall

The piece was premiered on December 14th, 2004, at the University of Alberta

6.4 Summary of Contributions

In this project, a mapping was devised between timbre and colour, as well as between chords and colour. The mappings were used to compose an audio-visual performance piece that conveyed a story through visual metaphor.

The music visualization application described in this chapter:

- Facilitated visual composition by creating a visual palette corresponding to the harmonic structure of the Circle of Fifths and the vowel sounds used in sung vocalization
- Allowed a performer to control responsive visualization in an artistic setting
- Told a story using dynamic audio-visual media

This system achieved both the artistic and technical goals stated in the introduction to this Chapter. The story of the artist and her ability to harness performance anxiety in order to produce a compelling performance was told through the visual imagery used in the performance. The technical goal of producing an interactive system that an artist could use successfully in a live performance setting was also achieved.

In addition to having been performed in concert at the University of Alberta, the piece is routinely performed in the laboratory in order to show visitors how visualization can be used for artistic purposes. It was also performed during several media interviews, including a live performance on CBC Radio.

It is my hope that the responsive video system used to create *Deep Surrender* could be expanded to use longer video sequences and additional vocal interaction mechanisms so that it could be installed in a museum or aquarium setting where visitors could interact with the jellyfish visualization for entertainment purposes.

Chapter 7

Visualizing Music Inside an Immersive Environment

This chapter discusses the technical aspects of an artistic visualization that I am currently developing based on Gaston Leroux' *The Phantom of the Opera* [17]. *The Phantom of the Opera* is the story of a disfigured musical genius who lives in a fantastical secret lair under the Paris Opera House. I am using an immersive space to transport my audience into the *Phantom*'s lair, where they can use music to interact with the space and explore the virtual environment. The artistic components of the installation are not yet complete, but the technical framework is in place to support the project's development.

The *Phantom* simulation uses the previously described Musical Perception Filter Layer (see Chapter 4 of this thesis) to interact with a responsive environment created using the visually programmed virtual reality visualization engine, Virtools [8].

7.1 Virtual Reality Applications in Immersive Environments

Virtools is a virtual reality development environment which allows applications to be displayed in immersive stereoscopic environments like the ones found in the Advanced Man-Machine Interface Laboratory's VizRoom (see Figure 7.1) and the

Figure 7.1: The University of Alberta VizRoom

Banff Centre's ARTLab.

These immersion rooms consist of three 10 foot by 10 foot rear-projected stereoscopic displays. A user stands inside the 3 walled enclosure, and perceives convincing three-dimensional life-sized imagery. Immersive environments facilitate a user's suspension of disbelief, as the realism of the imagery makes the virtual reality environment more believable.

Virtools is designed for use in immersive visualization laboratories and art studios. Virtools' VRPack add-on provides support for stereoscopic immersive environments and suitable interaction devices like trackers or joysticks. The Virtools engine communicates with these devices using VRPN [35]. Any input device supported by VRPN can be easily integrated into a Virtools simulation.

7.2 Virtools

Dassault Système's Virtools development environment is a visual programming environment (see Figure 7.2) specially designed to facilitate the creation of virtual reality applications [8]. Much like Max/MSP, Virtools allows designers to describe their applications graphically, using objects and connectors to visualize data flow.

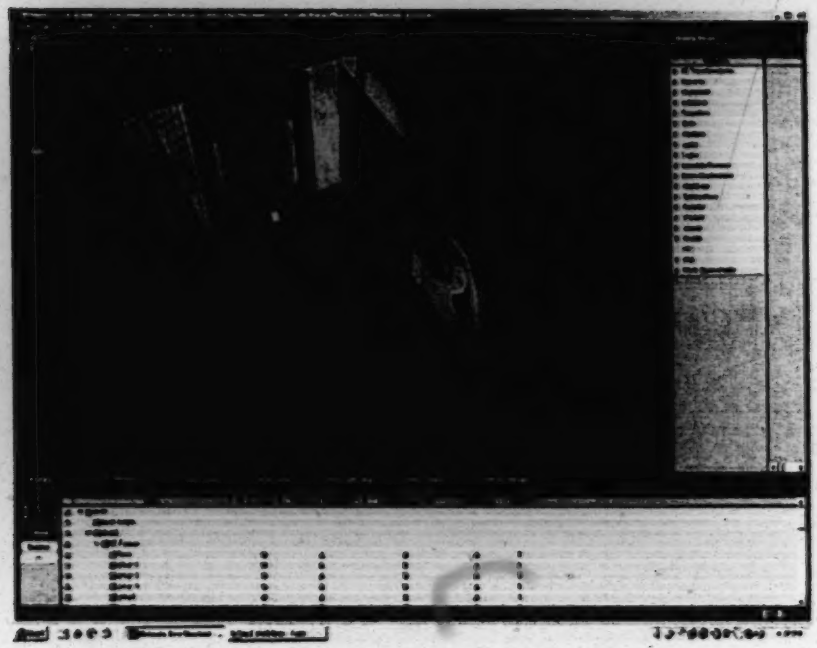


Figure 7.2: The Phantom of the Opera Virtools Project

Virtools can be used to create complex and visually appealing immersive virtual experiences. It is used to create installations in world-class artistic production environments like Alberta's Banff Centre.

7.2.1 The Virtools Development Environment

Virtools' authoring tools allow developers and designers to create VR applications visually, linking Behavior Building Blocks to one another in order to specify the flow of data within the application.

The Virtools Behavior Engine

Data processing inside the Virtools environment is done via the Behavioral Engine. *Behaviors* are functionalities that can be applied to *Behavioral Objects* in the Virtools environment. Behavioral Objects include 3D objects, cameras, lights, and sounds.

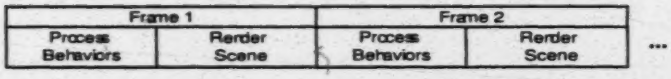


Figure 7.3: The Virtools Process Loop (adapted from [30], p. 97)

All events within the simulation are controlled via Behaviors. Examples of Behaviors that can be applied to Behavioral Objects include collision detection routines that can be applied to 3D objects, colour effects that can be applied to lights, and interaction techniques that can be used to control virtual characters. These Behaviors are executed when a Virtools simulation is activated.

The execution of a Virtools simulation is controlled by the Virtools process loop (see Figure 7.3.) This loop, known as a *frame*, runs continuously while a simulation is operational. Each frame contains two stages:

- All Behaviors are executed, and their effects on Behavioral Objects determined
- The simulation is rendered, displaying the updated Behavioral Objects

The connections between Virtools Behaviors and Behavioral Objects are defined visually, using the drag-and-drop Virtools Authoring Environment.

The Virtools Authoring Environment

The Virtools Authoring environment allows simulations to be defined by creating *scripts* that describe the objects in a scene and the Behaviors that should be applied to them. Scripts are visually programmed by connecting Behaviors to their associated Behavioral Objects.

Figure 7.4 illustrates a simple Virtools script which connects a virtual character object to the Building Blocks which allow her animations to be controlled by keyboard input. The *Character Controller*, *Keyboard Controller*, and *Character Keep On Floor* Behaviors operate upon the the Behavioral Object, *Eva*, to which they are

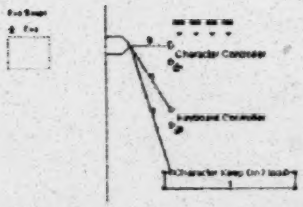


Figure 7.4: An Example of Visual Programming in Virtools [30], p. 143

attached.

Custom-Creating Building Blocks Using The Virtools SDK

Virtools authors may not need to manually code any part of the application if the extensive Virtools Building Block library contains all the features and Behaviors they need. If they do need to custom-create Building Blocks, Virtools makes this possible via a simple SDK that allows developers to code new Behaviors in C++.

This SDK is comprised of a simple set of wrapper classes that can be compiled in Visual Studio. New Building Blocks are defined by determining their input and output parameters, and the functionality that takes place each time the Building Block is executed.

Building Blocks are compiled as DLLs. Once the DLL is moved to the directory where Virtools Building Blocks are stored, it can be selected inside the Virtools Authoring Environment and used inside scripts to control Behavioral Objects.

Importing and Rendering 3D Data in Virtools

Virtools allows models and animations to be easily imported from 3D modelling software packages like Maya or 3D Studio Max. Plugins installed into the modelling software allow 3D data to be saved in the native Virtools format, ready to be drag-and-dropped into a Virtools project. The *Phantom of the Opera* set[1] seen in

[1]The textures used in the model were acquired from KTN 3D Models and Textures Website – http://www.ktn3d.com/

Figure 7.2 was imported in such a way.

Imported models are rendered by the Virtools renderer. The Virtools renderer allows designers to specify camera positions, dynamic lighting effects, and particle effects. If the Virtools renderer is not sufficient for a particular application, a customized renderer may be created and substituted.

7.3 The *Phantom of the Opera* Visualization

I am creating A *Phantom of the Opera* [17] installation that can be displayed in the immersive Virtools environment. The simulation is responsive to vocalization and keyboard playing. This section describes the technical aspects of the installation.

7.3.1 Musical Control

To enable musical interactivity within the Virtools simulation, it was required to connect the Virtools simulator with the Musical Perception Filter Layer described in Chapter 4 of this thesis. The Musical Perception Filter Layer extracts and transmits information from live musical input. Once received within the Virtools environment, this extracted information was linked to Virtools Building Blocks in order to control aspects of the Virtools simulation.

Using a customized building block, `MusicController`, which was built in Visual Studio using the accessible Virtools SDK, the Virtools engine was able to connect (via VRPN) to the Musical Perception Filter Layer and receive data.

Integration of Musical Control into the Virtools Process Loop

`MusicController` was implemented as a Virtools Behavior, and attached to the camera rendering the scene. Each time the processing loop iterated, the `MusicController` Behavior was executed.

The `MusicController` received updated musical feature data (pitch, amplitude, timbre and chord data) from the Musical Perception Filter Layer each time

it was triggered by the Virtools processing loop. This data could then be accessed inside the Virtools simulation via a set of parameters that output the musical feature data from the `MusicController`.

7.3.2 Audio Visual Mapping Example

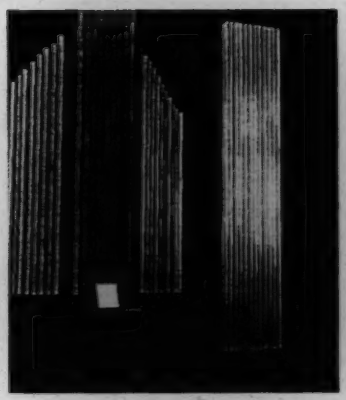
In order to experiment with visualizing sound inside immersive space, a visualization metaphor was created that uses particle clouds to visualize voice. The clouds are created and manipulated using Virtools' particle generation engine.

`MusicController`'s pitch and amplitude output parameters were connected to the colour and size input parameters of a Virtools particle emitter in order to allow the particle properties to be affected by the user's voice. When the singer's pitch changed, the particles' colour changed. When the singer's loudness increased, the size of the particles in the cloud increased.

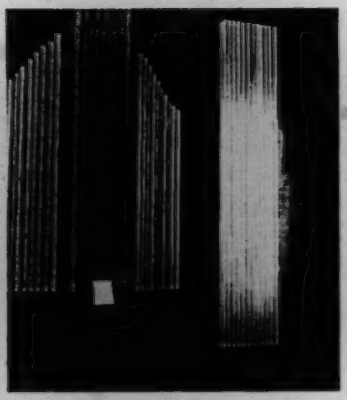
Mapping Pitch to Colour

It was desirable that the incoming pitches be mapped to particle cloud colours in such a way that the entire colour spectrum could be visualized if a singer sang every pitch encompassed by her range. Since I am a soprano, and was devising the system for my own artistic creation, I attempted to scale the system for my own vocal range (the three octaves from F below middle C (MIDI pitch 53) to the F above high C (MIDI pitch 89). The range specification would of course have to be customized for each voice type – a baritone's range would be much different than mine.

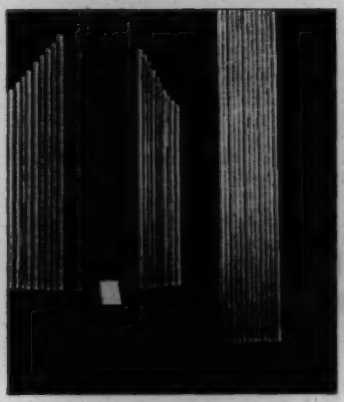
Since the target range contained 36 discrete pitches, incoming pitches were assessed relative to their position on this range, and scaled to a floating point number between 0 and 1. This was used as a hue input to a colour selector that was connected to the particle colour input. Low pitches produced particles that were a green-yellow colour, while the particles produced by high pitches looked vibrantly red.



(a) C4 (middle C) sung softly



(b) C5 sung at a moderately loud volume



(c) C6 (high C) sung loudly

Figure 7.5: Visualizing Pitch and Amplitude with Particle Fog

Mapping Amplitude to Size

The size of the particles in the particle cloud was also a definable parameter. Louder sounds were mapped to larger particles, and quieter sounds to smaller particles. This allowed the size of the particle cloud to be affected by the loudness or softness of the input vocalization.

Scaling was done on amplitude data in the same way as it was done with pitch: minimum and maximum vocal amplitudes were determined and mapped to the minimum and maximum particle sizes. All intermediary amplitudes were expressed in terms of their position within this range, and scaled to intermediary particle sizes.

Results

Figure 7.5 shows the resulting visualizations. The visible pitch range was tuned to my vocal range, so middle C sung softly was visualized as a small green-yellow cloud. The C an octave above that, sung at a moderately loud volume appeared as a slightly larger blue cloud. Singing high C at full volume produced a large red cloud.

The visual mapping took about two hours to create and customize. This customization time was spent choosing the appropriate particle parameters to modify (particle emitters have many parameters, modifying each of which produces different visual results) and fine-tuning the parameter scale factors. This is of course a simple example of responsive visualization, but it serves to illustrate how easily and effectively Virtools can be used to link parameterized sound data with imagery.

7.3.3 Future Development Plans

Virtools' ease-of-use and extensive Behavior libraries make it a valuable tool in the creation of artistic visualizations. The fact that a visual metaphor can be created and customized so rapidly makes it easy to experiment with numerous effects and parameterizations when creating an application. Being able to display the visual-

izations in an immersive environment increases their expressive and communicative potential.

I plan to continue developing the *Phantom* simulation inside Virtools, and combine Virtools' Behavior library with custom-created Behavior routines in order to make the simulation more complex. Each feature extracted by the Musical Perception Filter Layer should be mapped to visual parameters in the Virtools environment, resulting in a richly responsive visualization. When visitors to the simulation make sounds into the microphone or play upon the keyboard, features of the virtual environment will change in order to provide them with feedback.

I would like to experiment with goal-driven interactions (similar to the interactions that occur in Oliver *et al.*'s *Singing Tree* [23]) which reward participants for performing vocal improvisations within the responsive space by providing them with interesting visual feedback in response to their actions.

Integration of the ANIMUS Framework

Our laboratory is interested in porting Torres and Boulanger's ANIMUS Framework [36][37][38] to the Virtools platform in order to enable rapid prototyping of ANIMUS visualizations. I would like to integrate the musically responsive virtual characters discussed in Chapter 4 of this thesis into the Virtools simulator. The integration of the ANIMUS Framework and Virtools' immersive environment would be a valuable asset to artistic development in our laboratory.

7.4 Summary of Contributions

This chapter illustrates how the Musical Perception Filter Layer can interface with the visual programming environment provided by Virtools in order to facilitate the rapid prototyping of music visualization applications.

Although the visualization metaphor described in this chapter is simplistic, the

application serves to illustrate several features of our music visualization framework:

- The connection between the Musical Perception Filter Layer and the Virtools visualization engine allows a live musician to interact with an immersive simulation
- The immersive visualization and the associated musical feature extraction system are both visually programmed, allowing artists who may not have formal programming skills to participate in all aspects of the creation of music visualization applications
- Visualization metaphors may be rapidly developed, tested, and modified using the Virtools Authoring Environment

Now that the technical framework supporting development of music visualization applications inside the Virtools environment is in place, we look forward to finishing our *Phantom of the Opera* visualization, and continuing to develop artistic works inside the immersive environment.

Chapter 8

Conclusion

This thesis has presented a music visualization system which operates in a distributed fashion, facilitating easy re-use of the musical analysis module contained in the Musical Perception Filter Layer.

8.1 Music Visualization System Summary

The Musical Perception Filter Layer was implemented in the musical development environment Max/MSP [7]. Pitch, amplitude, timbral information and chord data were used to interact in real-time with aspects of a virtual environment.

Three experimental examples were implemented in order to show how the features extracted by the musical analysis module could be mapped to different visualization metaphors.

Interactive ANIMUS virtual characters [37] were used to illustrate the emotive capacities of music through visible behaviours that illustrated emotional responses to music. The cognitive system used to trigger these behaviours was consistent with Deryck Cooke's research correlating melody and intention [5].

Cycling '74's Jitter environment [6] was used to create responsive video streams which responded to vocal and keyboard input. This visualization was used to create a multimedia performance piece, *Deep Surrender*.

The Virtools visual programming environment was used to create a visualization

which can be performed in an immersive virtual space. This visualization used particle dynamics to illustrate vocal performance.

8.2 Expressing Musical Emotion through Visualization

With each music visualization application we created, we augmented live musical performance using visual imagery to represent the performance's emotive content. Each visualization interpreted emotive content from different musical feature data, and illustrated it using a different visual metaphor.

8.2.1 Extracting Musical Descriptors

The musical feature data extraction system used to implement these visualizations parameterized live music into a series of perceptual parameters. Information about vocal pitch, loudness, and timbre, as well as chord data from the digital piano was available to the visualization engines.

Each example visualization illustrated emotion conveyed by different musical descriptors:

- The ANIMUS virtual character simulation was responsive to sung pitches, encoded within a tonal context relative to the key signature of the sung melody
- The responsive video environment reacted to aspects of the vocalist's timbre
- The particle cloud simulation inside the immersive space was responsive to the vocalist's pitch and loudness

8.2.2 Interpreting Emotional Content in Musical Performance

Different methods of associating the extracted musical feature data with emotive content were used in each experimental implementation:

Responsive Virtual Characters

The responsive virtual character simulation interpreted emotional content in a musical piece using Cooke's metric correlating composer's emotional intentions and the tonal contextualities present in a melody line [5].

To interpret emotional content from live music using this strategy, the incoming melodies were encoded within the tonal context of their key signature, and each note in the melody was assigned an emotive capacity. As the melody progressed, the cumulative impact of the emotive association of each note affected the emotional state of a virtual character.

Responsive Video

In the *Deep Surrender* responsive video project, the timbre and duration of a singer's vocalizations was interpreted to be indicative of her emotional state. If the singer made breathy staccato sounds, the emotive content was interpreted as 'hesitant' and 'nervous'. If she held focused and prolonged pitches, this conveyed her 'confidence' and 'elation'.

Chord data was also used in this implementation to convey mood. Minor chords were interpreted as 'ominous', while major chords were interpreted as 'optimistic'.

Responsive Particle Clouds

The Virtools simulation illustrates a simple mapping between the vocal intensity (conveyed through pitch and volume) and emotional intensity. If a singer makes loud and high noises, this is interpreted as being 'intense', while low and soft noises are interpreted as being more 'relaxed'.

8.2.3 Illustrating Emotion Through Responsive Music Visualization

Each experimental implementation illustrated emotional content in the live performance using different visual metaphors.

Responsive Virtual Characters

The ANIMUS character visualization illustrated the emotion contained in sung melodies through the visible responses of a virtual character. The anthropomorphic Alebrije character could transition between 'happy' and 'sad' postures to reflect the emotive content in the melodic passages. This visualization metaphor was quite straightforward, since the emotive content of Alebrije's poses was easily recognizable, and the emotive content of the tonal music used as input to this system was understandable to the viewing audience. The animation of a cringing animated lizard who hides his eyes and slumps to the ground in response to hearing the haunting strains of "Greensleeves" is accessibly evocative of 'sadness'. The mappings used in the ANIMUS character visualization correlated 'sad' music with 'sad' character poses in a direct way.

Responsive Video

The visual metaphors used in the responsive video representation mapped musical emotion to visual imagery in a more indirect fashion. They mapped chord data, and vocal timbre to the colour balance and image layering effects that were used to manipulate a responsive video.

The chords that were used in the *Deep Surrender* composition were associated to various colours. Certain colour palettes (icy pinks, pale blues and light greens) were selected to illustrate phases of the piece which conveyed 'fear', and brighter colours (bold reds, bright greens and vivid blues) were chosen in order to convey 'elation'.

Adding an improvisational component to the pre-defined colour and chord progressions, *Deep Surrender* used colour and chroma-keyed image layering to illustrate the singer's vocal timbre as the piece was performed. The singer could watch the monitor during each live performance, and modulate her vowel sounds in order to improvisationally control the colour balance and chroma-keyed imagery in the video playback. Using these audio-visual mappings to choreograph the playback of pre-rendered visual footage, the singer could illustrate the different emotional phases of the piece by manipulating the colour intensity of the visual imagery appropriately during each section.

Small hesitant sounds produced flashes of pale blue and green imagery during the phases of the piece when the singer's anxiety was being illustrated, while more confident and full-voiced vocalization produced vibrant reds and purples in the later stages of the performance when her elation was evident.

Responsive Particle Clouds

In the Virtools simulation, the loudness and pitch of vocal sound was mapped to a responsively sized and coloured particle cloud. Louder vocalization produced larger clouds, while quieter visualization produced smaller clouds. Different pitches produced different coloured clouds.

Using this type of metaphor, the emotional intensity of vocalization is illustrated via the size and colour choice associated with each type of sound. Shrill and loud sounds were mapped to dramatically large and bright particle clouds, while low quiet sounds were mapped to less vibrant particle clouds, that were smaller and composed of duller colours.

8.3 Future Artistic Direction

While the *Deep Surrender* multimedia performance is a completed work, the other visual metaphors (the virtual character and the immersive environment) are too simplistic to be artistically cohesive. In the near future, a collaboration is planned with a digital artist who plans to produce an extensive library of animation routines for use with the ANIMUS virtual character engine. We plan to adapt the ANIMUS character framework to the visual programming environment provided by Virtools, in order to aid in rapid character development.

With our artist's expressive set of character animations to use as a palette, we plan to create an installed work whereby musicians and audience members can use their voices to communicate with a fluidly animated virtual character inside a compelling immersive space.

8.4 Future Applications

Although this system was primarily devised as a tool for live multimedia performance, the system could be used to assist in other tasks as well. The system could be a useful learning tool, providing visual feedback to vocalists who wish to develop or refine their singing techniques.

In the speech therapy realm, visual feedback could help patients visualize subtleties in vocal pitch, amplitude or timbre that they may not be able to isolate from auditory feedback. The entertaining nature of visualization could help make speech therapy exercises more interesting for young patients.

Visualizing vowel production could help people improve pronunciation when learning a second language. Visual feedback could aid learners in mastering the specific nuances of vowel sound and intonation.

In order implement these non-performance-oriented vocal visualizations it would

of course be necessary to develop specialized visual metaphors for the system, and to refine and extend the vocal extraction system, particularly in the area of timbral analysis.

8.5 Summary of Contributions

We theorized that this application could provide a distributed framework for the rapid development of applications that use visual metaphors to express aspects of musical performance.

In order to create such a framework, we addressed the following issues:

- The extraction of musical feature data from acoustic and MIDI input was done inside the visual programming environment Max/MSP
- Musical feature data extraction was kept separate from visualization in order to facilitate modularity
- Musical features were organised inside a representation scheme consistent with tonal music theory
- Visual metaphors were created to express musical content in artistic ways:
  - Emotion in music was visualized through the responses of an ANIMUS character
  - An audio-visual performance piece illustrated vocal timbre and chord relationships through the use of colour manipulation and responsive video processing created inside the Jitter visual programming framework
  - Vocal pitch and loudness were visualized using particle clouds in an immersive setting using Virtools' visual programming system

This framework has provided a way to generate multiple visualization metaphors to express extracted musical feature data. The virtual character and particle cloud visualizations can potentially be used to create artistic pieces, and the responsive video production has already been used in live performance.

Care has been taken to develop this system in ways that facilitate rapid development of creative visualizations and collaborative work between artists and programmers. Max/MSP, Jitter, and the Virtools environment can be visually programmed, making possible the rapid prototyping of visual metaphors. The fact that the system allows visualizations and musical feature extraction to be conducted in visually programmed environments makes it more accessible to artists who may have no formal training in computer programming. Although developing visualizations with the ANIMUS environment is more complex, ANIMUS is designed with the idea of task delegation in mind, allowing artists and programmers to work alongside one another to develop creative works.

I look forward to continuing to use this system to create artistic works that combine live music and responsive visualization.

Bibliography

[1] Ronald N. Bracewell. *The Fourier Transform and Its Applications. Second Edition, Revisited.* McGraw-Hill Book Company, 1986.

[2] Murray Campbell and Clive Greated. *The Musician's Guide to Acoustics.* New York: Schirmer Books, 1987.

[3] Marc Cardle, Loic Barthe, Stephen Brooks, and Peter Robinson. Music-driven motion editing: Local motion transformations guided by music, June 2002.

[4] Catherine Ikam and Louis-Francois Fléri and Music by Pierre Charvet. Elle et la Voix, 2000.

[5] Deryck Cooke. *The Language of Music.* New York: Oxford University Press, 1959.

[6] Cycling '74. Jitter, 2004.

[7] Cycling '74. Max/MSP, 2004.

[8] Dassault Syst èmes. Virtools, 2005.

[9] Diana Deutsch and J. Feroe. The internal representation of pitch sequences in tonal music. *Psychological Review*, 88:503–522, 1981.

[10] Electronic Arts. The Sims, 2000.

[11] M. Goto. An audio-based real-time beat tracking system for music with or without drum-sounds, 2001.

[12] Tae hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Graph.*, 22(3):392–401, 2003.

[13] T. Jehan, T. Machover, and M. Fabio. Sparkler: An audio-driven interactive live computer performance for symphony orchestra. In *Proceedings of the International Computer Music Conference*. International Computer Music Association, 2002.

[14] T. Jehan and B. Schoner. An audio-driven perceptually meaningful timbre synthesizer. In *Proceedings of the International Computer Music Conference*, pages 381–388. International Computer Music Association, 2001.

[15] S. Koelsch, T. Gunter, A.D. Friederici, and J. Schroger. Brain indices of music processing: "nonmusicians" are musical. *Cognitive Neuroscience*, 12:520–541, 2000.

[16] Carol L. Krumhansl. The geometry of musical structure: a brief introduction and history. *Comput. Entertain.*, 3(4):3–3, 2005.

[17] Gaston Leroux. *The Phantom of the Opera*. New York: Bobbs-Merrill, 1911.

[18] Golan Levin and Zachary Lieberman. In-situ speech visualization in real-time interactive installation and performance. In *Proceedings of The 3rd International Symposium on Non-Photorealistic Animation and Rendering*, pages 7–14. ACM Press, 2004.

[19] Masataka Goto and Yoichi Muraoka. Interactive Performance of a Music-Controlled CG Dancer. http://staff.aist.go.jp/m.goto/PROJ/ip-j.html.

[20] R. McNab, L. Smith, and I. Witten. Signal processing for melody transcription. In *Proceedings of the 19th Australasian Computer Science Conference*, pages 301–307, 1996.

[21] Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, Clare L. Henderson, and Sally Jo Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Digital Libraries*, pages 11–18, 1996.

[22] Eric Metois. *Musical Sound Information: Musical Gesture and Embedding Synthesis*. PhD thesis, Massachusetts Institute of Technology, 1996.

[23] William Oliver, John Yu, and Eric Metois. The Singing Tree: design of an interactive musical interface. In *DIS '97: Proceedings of the conference on Designing interactive systems : processes, practices, methods, and techniques*, pages 261–264. ACM Press, 1997.

[24] Jack Ox. 2 performances in the 21st Century Virtual Color Organ. In *Proceedings of the fourth conference on Creativity & Cognition*, pages 20–24. ACM Press, 2002.

[25] A.D. Patel, E. Gibson, J. Ratner, M. Besson, and P.J. Holcomb. Processing syntactic relations in language and music: An event-related potential study. *Cognitive Neuroscience*, 10:717–733, 1998.

[26] John R. Pierce. *The Science of Musical Sound*. Scientific American Books, 1983.

[27] Christopher J. Plack. *The Sense of Hearing*. Lawrence Erlbaum Associates, Inc., 2005.

[28] M. Puckette, T. Apel, and D. Zicarelli. Real-time audio analysis tools for Pd and MSP. In *Proceedings of the International Computer Music Conference*, pages 109–112. International Computer Music Association, 1998.

[29] M. Puckette and A. C. Lippe. Score following in practice. In *Proceedings of the International Computer Music Conference*, pages 182–185, 1992.

[30] Virtools SA. *Virtools Dev User Guide*. 2004.

[31] Eric Singer, Athomas Goldberg, Ken Perlin, Clilly Castiglia, and Sabrina Liao. Improv: Interactive improvisational animation and music. In *Proceedings of the International Society for the Electronic Arts (ISEA) Annual Conference*, 1996.

[32] Johan Sundberg. *The Science of the Singing Voice*. Northern Illinois University Press, 1987.

[33] Robyn Taylor, Pierre Boulanger, and Daniel Torres. Visualizing emotion in musical performance using a virtual character. In *Proceedings of the Fifth International Symposium On Smart Graphics*, pages 13–24. Springer LNCS, 2005.

[34] Robyn Taylor, Daniel Torres, and Pierre Boulanger. Using music to interact with a virtual character. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 220–223, 2005.

[35] Russell M. Taylor II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T. Helser. VRPN: A device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61. ACM Press, 2001.

[36] Daniel Torres. The ANIMUS Project: A framework for the creation of synthetic characters. Master's thesis, University of Alberta, 2003.

[37] Daniel Torres and Pierre Boulanger. The ANIMUS Project: a framework for the creation of interactive creatures in immersed environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 91–99. ACM Press, 2003.

[38] Daniel Torres and Pierre Boulanger. A perception and selective attention system for synthetic creatures. In *Proceedings of the Third International Symposium On Smart Graphics*, pages 141–150, 2003.

[39] Tod Winkler. *Composing Interactive Music*. Cambridge: MIT Press, 1998.